VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Návrh a realizace systému pro sběr a vizualizaci senzorických dat pro internet věcí

Design and Implementation of a System for Collecting and Visualizing Sensory Data for the Internet of Things

Bc. Michal Čepelák, DiS.

Diplomová práce

Vedoucí práce: Ing. Radim Hercík, Ph.D.

Ostrava, 2021

Poděkování

Rád bych poděkoval Ing. Radimu Hercíkovi, Ph.D. za odbornou pomoc a konzultaci při vytváření diplomové práce. Dále bych chtěl poděkovat celému kolektivu společnosti Home Technologies s. r. o. za to, že mi umožnili podílet se s nimi na tomto projektu.

Abstrakt

Diplomová práce se zabývá návrhem a realizací systému pro sběr a vizualizaci senzorických dat pro společnost Home Technologies. Webové rozhraní na webhostingu poskytovatele Wedos bylo realizováno ve skriptovacím jazyce PHP s prvky Javascriptu a kaskádových stylů CSS. Pro uchování senzorických dat byla využita databáze MariaDB obsluhovaná dotazovací třídou MySQLi jazyka PHP. Práce se v závěru věnuje testování a ověření funkcí systému.

Klíčová slova

Internet věcí, PHP, MySQLi, MariaDB, Firebase Cloud Messaging, HTML, POST, GET

Abstract

The diploma thesis deals with the design and implementation of a system for the collection and visualization of sensory data for the company Home Technologies. The web interface on the web hosting provider Wedos was implemented in the PHP scripting language with Javascript elements and CSS cascading style sheets. The MariaDB database served by the PHP MySQLi querying class was used to store sensory data. At the end, the work deals with testing and verification of system functions.

Key Words

Internet of Things, PHP, MySQLi, MariaDB, Firebase Cloud Messaging, HTML, POST, GET

Obsah

Seznam použitých symbolů a zkratek

API	Application Programming Interface
ASP	Active Server Pages – skriptovací jazyk
BLE	Bluetooth Low Energy
BME	Název zařízení pro měření teploty a relativní vlhkosti společnosti Home Technologies
FCM	Firebase Cloud Messaging
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEEE	the Institute of Electrical and Electronics Engineers
IoT	Internet of Things – Internet věcí
ITU	Mezinárodní telekomunikační unie
JSON	JavaScript Object Notation
KPI	Key Performance Indicator – klíčové ukazatele výkonnosti
MQTT	Message Queuing Telemetry Transport
NoSQL	Not only Structured Query Language nebo Non SQL
PBX	Název senzoru poštovní schránky společnosti Home Technologies
PHP	Hypertext Preprocessor – skriptovací jazyk
PoE	Power over Ethernet
QoS	Quality of Service
REST	Representational State Transfer
RFC	Request for Comments – dokumenty popisující internetové standardy
RLY	Název zařízení s relé akčním členem společnosti Home Technologies
RSSI	Received Signal Strength Indication – indikátor síly přijímaného signálu
SQL	Structured Query Language
s.r.o.	Společnost s ručením omezeným
SSID	Service Set Identifier – identifikátor bezdrátové sítě wifi
tzv.	takzvaně, takzvaný
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTFB	Time To First Byte
URL	Uniform Resource Locator
USB	Universal Serial Bus
XMPP	Extensible Messaging and Presence Protocol

Seznam ilustrací

Seznam tabulek

Úvod

V současnosti je internet věcí spolu s průmyslem 4.0 stále větším trendem a neustále roste počet jeho uživatelů a zařízení připojených k těmto standardům. Lze očekávat, že zájem o vzdálenou správu chytrých zařízení v budoucnosti ještě poroste.

Pro zpracování diplomové práce bylo zvoleno téma návrh a realizace systému pro sběr a vizualizaci senzorických dat pro internet věcí. Práce je realizována pro firmu Home Technologies s.r.o.

Cílem diplomové práce je navrhnout strukturu databáze pro ukládání dat z jednotlivých senzorů a dalších informací o uživatelích, senzorech a výrobním procesu zařízení. Dalším krokem po dokončení návrhu je databázi zrealizovat, otestovat a ověřit funkce a parametry systému pro sběr a vizualizaci senzorických dat pro internet věcí.

Práce je rozdělena na dvě části, a to na teoretickou a praktickou část. V úvodu práce je popsána společnost Home Technologies s.r.o. a její vyvíjená zařízení. Teoretická část diplomové práce je dále zaměřena na rozbor problematiky sběru dat na platformě internet věcí. Je zde popsán internet věcí a technologie, které využívá pro přenos dat. Jsou zde uvedeny komunikační protokoly použitelné pro internet věcí. Další část práce obsahuje rešerši dostupných systémů pro ukládání dat ze senzorů pro IoT, kde jsou popsány druhy databázových systémů a jejich srovnání.

Následující kapitola je zaměřena na návrh systému sběru a vizualizace dat. Tato kapitola se zabývá výběrem webhostingu a databáze. Součástí této kapitoly je i návrh jednotlivých databázových tabulek a odhad jejich velikosti.

Nejdůležitější částí práce je samotná realizace systému. Kapitola je rozdělena na několik oddílů obsahujících průběh vývoje systému, například proces vytvoření webového rozhraní a popis jeho základní struktury. Dále je zde uveden princip příjmu a zpracování dat v systému a omezení přístupu k webu pomocí registrace a přihlašování uživatelů. Jsou zde popsány důležité součásti webu pro správu připojených zařízení, zpracování oznámení a vizualizaci shromážděných dat pomocí tabulek a grafů.

V závěru práce je provedeno testování a verifikace funkcí a parametrů systému. Tato část obsahuje srovnání odhadu velikosti databáze se skutečnou velikostí po roce provozu. Je zde realizováno měření délky vykonávaných skriptů a popis ověření funkčnosti uživatelského rozhraní.

Nakonec jsou zhodnoceny dosažené výsledky práce.

1 Home Technologies s.r.o.

Diplomová práce je realizována pro firmu Home Technologies. Společnost Home Technologies byla zapsána do obchodního rejstříku dne 28. července 2020 a je vedena u Krajského soudu v Ostravě jako společnost s ručením omezeným. Společnost se zabývá vývojem a výrobou chytrých zařízení pro internet věcí. Hlavním impulsem pro založení podniku byla snaha o usnadnění každodenních činností, kterými lidé tráví příliš mnoho času. Díky produktům firmy Home Technologies je možné zautomatizovat některé rutinní činnosti a udělat si tak život pohodlnější. Příkladem může být senzor umístitelný do poštovní schránky, který detekuje přítomnost dopisu či jiné pošty ve schránce. Senzor upozorní svého uživatele na příchozí poštu podobně jako emailová schránka [22].

Dalším chytrým zařízením vyvíjeným firmou Home Technologies je výrobek označený jako BME. BME je usb dongle o velikosti většího usb flash disku, svůj název dostal podle senzoru BME280, kterým je osazený. Toto zařízení slouží k měření teploty, relativní vlhkosti a atmosférického tlaku. Naměřené hodnoty jsou pomocí wifi odesílány na server Home Technologies a dále zpracovávány.

Do portfolia chytrých zařízení vyvíjených a vyráběných společností Home Technologies lze mimo senzorů zařadit také relé akční člen RLY a zobrazovací jednotku Pixie. RLY je bezdrátové dvoukanálové wifi relé napájené 230 V ze sítě, obsahující dvě nezávislé přepínací relé ovládané přes wifi, nebo i lokálně, připojením vypínače k ovládací svorkovnici.

Všechna zařízení firmy Home Technologies se v rámci místní wifi sítě spárují a mohou spolu komunikovat a sdílet informace. Příkladem takového spojení může být použití zobrazovací jednotky Pixie, která takto získané hodnoty může okamžitě zobrazovat pomocí vestavěné barevné LED matice 6x24 pixelů. Kromě zobrazení přímo získaných hodnot ze spárovaných zařízení nabízí jednotka Pixie zobrazení digitálních hodin, informací o počasí, funkci domácího zvonku se světelnou indikací zvonění, a mnoha dalších zajímavých dat z internetu.

Data získaná ze všech svých zařízení ukládá společnost Home Technologies s.r.o na vlastní zabezpečené databázi a nabízí tak uživatelům možnost zobrazení a další práce s naměřenými daty pomocí databázového systému s webovým rozhraním MARS (Modern Access Remote System). Diplomová práce se zabývá právě návrhem a realizací databázového systému a webové aplikace MARS. Vytvoření vlastního systému MARS pro sběr, uchování a vizualizaci dat je výhodnější než využívání zpoplatněných a často uzavřených systémů třetích stran. Vlastní řešení navíc umožňuje prakticky neomezené možnosti rozšíření funkcionality podle aktuálních požadavků trhu. Data uživatelů jsou uloženy v České republice a vztahují se na ně české zákony.

Od počátku roku 2021 firma usilovně pracuje na integraci svých zařízení do stávajících systémů pro chytré domácnosti jako jsou Google Home, Home Assistant, Apple Homekit atp. a přidává do svých zařízení také podporu protokolu MQTT.

Home
Technologies
Inside matters

www.hometechnologies.cz

Obrázek 1 Logo Home Technologies

2 Rozbor problematiky sběru a zpracování dat ze senzorů pro IoT

Tato kapitola diplomové práce obsahuje seznámení s platformou internet věcí, rozbor problematiky sběru dat pomocí IoT zařízení a stručný popis technologií k přenosu dat v sítích internetu věcí. Dále je zde zahrnut popis komunikačních protokolů, které je možné použít pro zařízení v internetu věcí.

2.1 Internet věcí

Internet věcí je známý především pod anglickým názvem Internet of Things zkráceně IoT. Základním principem internetu věcí je připojení běžně používaných zařízení k internetu. Zařízení připojená do sítě internetu věcí je možné vzdáleně spravovat a ovládat. Připojení k internetu věcí umožňuje stále více zařízení od teplotních senzorů až po lednice, robotické vysavače a televizory. Do sítě IoT lze připojit i další předměty v domácnosti, které tuto možnost obvykle nenabízí, a to pomocí různých druhů nabízených rozšíření. Příkladem může být poštovní schránka doplněná o jednotku se senzorem přiblížení a připojením k internetu, nebo spínání elektroniky pomocí IoT reléového modulu na přívodním kabelu.

K přenosu dat mezi zařízením a sítí internetu věcí je možné použít několik druhů přenosových médií. Lze využít drátová i bezdrátová přenosová média. V případě drátového připojení je vhodné využít vedení běžně používané v internetových sítích, a to nejčastěji pomocí síťového kabelu s konektory RJ45. Pro bezdrátové připojení se využívá wifi nebo komunikační technologie určené pro propojení nízkovýkonových zařízení například Bluetooth, Zigbee a podobně.

Tabulka 1 Srovnání přenosových technologií používaných v IoT.

Technologie přenosu	Dosah	Přenosová rychlost	Počet zařízení v jedné síti
Datový kabel UTP	až 100 m	1 – 10 Gbit/s	-
Wifi	10 – 100 m	Až 9,6 Gbit/s [10]	25 až 250
Bluetooth	10 – 100 m	až 2 Mbit/s	až 7[1]
Zigbee	až 75 m	20 – 250 kbit/s	až 65535
LoRa	2 – 20 km	292 bit/s – 50 kbit/s	Teoreticky neomezený
Sigfox	3 – 50 km	až 100 bit/s	Teoreticky neomezený
NB-IoT	až 20 km	250 kbit/s	Teoreticky neomezený

[1] Až 7 zařízení současně komunikujících, teoreticky neomezený počet párovaných zařízení

2.1.1 Připojení pomocí datového kabelu

Pro připojení k internetu je vhodné využít síťové kabely typu UTP, případně stíněné síťové kabely s konektory RJ45, které se běžně používají v počítačových sítích. Maximální délka stíněných síťových kabelů bez přerušení může být až 100 metrů, což je pro domácnost a okolí domu plně dostačující.

Výhodou drátového přenosu pomocí síťových kabelů je stabilita připojení, menší pravděpodobnost rušení a možnost využití volných vodičů v kabelu pro napájení senzoru. Napájení senzoru po datovém síťovém kabelu se nazývá PoE, což je zkratka pro Power over Ethernet. Odpadá tak nutnost přivádět k senzoru samostatný napájecí kabel. Nevýhodou drátového připojení je složitější konstrukce senzoru a potřeba přivést kabel až k senzoru.

2.1.2 Wifi

Přenos pomocí wifi je nejčastější druh bezdrátového připojení k internetu. Označení wifi v informatice zahrnuje několik standardů IEEE 802.11 popisujících bezdrátovou komunikaci v počítačových sítích. Tato technologie využívá bezlicenční frekvenční pásma 2,4 GHz a 5 GHz. Dosah wifi sítě se pohybuje od 10 m v uzavřeném prostoru až po 100 m v ideálních podmínkách. [5]

Výhodou použití wifi pro přenos dat mezi zařízeními IoT a internetem je fakt, že bezdrátové sítě wifi se nachází prakticky v každé domácnosti. Odpadá tedy nutnost vytvářet nové bezdrátové sítě. Nevýhodou komunikace přes wifi je velká energetická náročnost, která výrazně zkracuje výdrž v případě napájení z baterie.

2.1.3 Bluetooth

Bluetooth je bezdrátová technologie umožňující výměnu dat mezi různými zařízeními na krátké vzdálenosti. Většina produktů využívajících Bluetooth pracuje na bezlicenčním pásmu 2,4 GHz s maximálním dosahem připojení kolem 10 metrů, tento dosah se ještě zkracuje v případě překážek mezi vysílačem a přijímačem. Je možné pořídit i Bluetooth zařízení, které má vyšší vyzařovací výkon a dosah až 100 m ve volném prostředí. Pro připojení zařízení do internetu věcí s využitím Bluetooth je zpravidla nutné mít v dosahu centrální jednotku, která data odesílá dále do internetu pomocí jiných standardů, například wifi.

Výhodou použití Bluetooth je nízká energetická náročnost přenosu dat, zejména při použití tzv. BLE - Bluetooth Low Energy. Mezi další kladné vlastnosti patří možnost propojení více zařízení současně a také to, že přenos dat může být zabezpečený šifrováním.

2.1.4 Zigbee

Je to bezdrátová komunikační technologie splňující standard IEEE 802.15.4 primárně určená pro senzorové sítě a průmyslové aplikace. Zigbee je podobně jako Bluetooth určena pro přenos dat mezi nízkovýkonovými zařízeními s dosahem do 75 m. Pracuje v bezlicenčních pásmech 868 MHz, 902-928 MHz a 2,4 GHz. Zigbee je navrženo jako jednoduchá technologie umožňující vytvoření i rozsáhlejších bezdrátových sítí s přenosy menších objemů dat. Každé zařízení v síti Zigbee má svou unikátní adresu, pomocí které jej lze adresovat. Jedna síť může obsahovat až 65 535 zařízení.

2.1.5 LoRa

LoRa je zkratkou pro Long Range. LoRa umožňuje bezdrátový přenos malých objemů dat na velké vzdálenosti. Síť využívající technologii LoRa se nazývá LoRaWan a v České republice ji provozují především České radiokomunikace (CRA). V roce 2018 pokrývala více než 70 % obyvatelstva [6]. Dosah sítě LoRaWan je až 20 km ve volné krajině a 2-5 km v husté městské zástavbě [7]. Ačkoliv LoRa v Evropě využívá bezlicenční frekvenci 868 MHz, je její používání u provozovatele CRA zpoplatněno a omezeno počtem odeslaných a přijatých zpráv za den. Je možné vytvořit si i soukromou síť LoRaWan bez některých výše jmenovaných omezení.

2.1.6 Sigfox

Sigfox s 96% pokrytím populace České republiky patří mezi nejrozšířenější sítě pro internet věcí [8]. Síť využívá bezlicenční frekvenci 868 MHz a je primárně určená pro vzdálenější zařízení. Nabízí dosah signálu až 50 km ve volné krajině a 3-5 km v městské zástavbě. V základní nabídce může IoT zařízení vyslat 144 zpráv a přijmout 4 zprávy, což znamená jednu odeslanou zprávu každých 10 minut. Každé zařízení v síti Sigfox má unikátní 32bitové Sigfox ID, které má přiřazené od výroby. Všechny zprávy v síti jsou podepsány privátním klíčem a šifrovány [7]. Data ze zařízení v síti se ukládají do Sigfox Cloudu, kde je možné je dále zpracovávat nebo odesílat na klientské servery.

2.1.7 NB-IoT

Síť NB-IoT, jak napovídá její název, je vytvořená výhradně pro chytrá IoT zařízení. NB je zkratkou pro Narrow Band tedy úzké pásmo a budují ji mobilní operátoři. NB-IoT využívá stávajících LTE sítí v licencovaném pásmu, ze kterých je vyčleněno úzké pásmo signálu určené pro IoT zařízení. K vytvoření sítě NB-IoT postačí pouze aktualizace software stávajících základnových stanic a nemusí se instalovat nové vysílače. Vzhledem k tomu, že se jedná o podsíť LTE, musí zařízení v této síti obsahovat komunikační modul s vestavěnou SIM kartou nebo eSIM. Dosah signálu v síti NB-IoT je kolem 20 km a je teoreticky dostupný všude, kde je pokrytí sítí LTE [7]. Podle webových stránek poskytovatele (Vodafone) síť pokrývá 100 % plochy České republiky a 94 % populace signálem uvnitř budov [9].

2.2 Komunikační protokoly použitelné pro IoT

V současné době se nabízí hned několik různých standardů pro komunikaci mezi zařízeními internetu věcí. V této kapitole jsou tyto standardy představeny a srovnány.

2.2.1 HTTP

HTTP je obecně známá zkratka pro HyperText Transfer Protocol. Je to internetový protokol používaný pro komunikaci s webovými servery poprvé představený v roce 1991. Samotné HTTP neumožňuje zabezpečení dat, a proto je stále častěji nahrazováno zabezpečenou a šifrovanou variantou HTTPS (HyperText Transfer Protocol Secure).

Pro zobrazení webové stránky se nejčastěji používá tzv. HTTP GET požadavek, pomocí kterého je možné u serveru požádat o zobrazení webové stránky a současně k požadavku připojit i další parametry ovlivňující zobrazení dané stránky. GET požadavek se skládá z URL adresy požadované webové stránky, kterou lze přímo v adresovém řádku webového prohlížeče doplnit o další parametry. Parametry pro HTTP GET požadavek se od URL webu oddělují symbolem otazníku a jednotlivé parametry jsou mezi sebou odděleny symbolem ampersand. Výsledná adresa s parametry může mít například následující tvar:

https://www.hometechnologies.cz/db/frgtpass.php?mail=cep0030@vsb.cz

Po odeslání požadavku ve zmíněném formátu může požadovaná stránka pracovat s hodnotami odeslanými v URL adrese. V případě tohoto požadavku se uživateli zobrazí webová stránka obsahující předvyplněnou emailovou adresu pro obnovení zapomenutého hesla. Tento způsob odesílání dat webovým stránkám ale není bezpečný, jelikož parametry může kdokoliv přečíst z adresové řádky. Proto je bezpečnější používat další druh požadavku hypertextového přenosového protokolu a tím je HTTP POST. Požadavek POST pracuje na stejném principu skládání parametrů jako HTTP GET ale s tím rozdílem, že parametry jsou uloženy uvnitř těla POST požadavku, takže nejsou přímo viditelné v adresové řádce.

Požadavky GET i POST lze poté na serveru jednoduše vyčíst například pomocí jazyku PHP jako asociativní pole proměnných. Parametr z předchozího příkladu s HTTP GET požadavkem obsahujícím emailovou adresu je možné v jazyce PHP získat přímo jako proměnnou, kde název pole přijatých dat je vždy *$_GET[]*. Jako prvek pole k získání daného parametru je použit název proměnné zapsaný v URL adrese s dotazem, výsledný zápis v jazyce PHP k získání proměnné z HTTP GET požadavku je *$_GET["mail"]*. V případě čtení parametrů z POST požadavku je syntax stejný, ale název pole je v tomto případě *$_POST[]*.

Komunikace pomocí HTTP byla původně navrhnuta pouze pro komunikaci na modelu klient/server, kde klient vždy začíná komunikaci odesláním dotazu na server. HTTP tedy neobsahuje vestavěný mechanismus umožňující udržet stálé propojení mezi klientem a serverem ani není jednoduše možné vyslat data ze serveru klientovi bez toho, aby klient poslal požadavek jako první. Existuje několik možností, jak se lze alespoň částečně tomuto nedostatku vyhnout.

Cyklické dotazování

Pro zajištění alternativy stálého propojení klienta a serveru umožňujícího obousměrnou komunikaci je možné cyklicky vysílat HTTP GET nebo POST požadavky na server. Při každém dotazu si tak klient se serverem mohou měnit data. Metoda cyklického dotazování je velmi spolehlivé řešení komunikace mezi klientem a serverem, která je v případě použití HTTPS zabezpečena šifrováním. Nevýhodou tohoto spojení je relativně velká latence a zatížení serveru při vyšším počtu klientů.

Long Polling

Další možností, jak zajistit přenos dat ve směru ze serveru ke klientovi bez nutnosti čekání na zahájení nového spojení klientem, je tzv. HTTP Long Polling. Principem metody HTTP Long Polling je navázání klasického spojení klientem pomocí GET požadavku se serverem, který odkládá odpověď. Spojení tak zůstává navázáno a v této době může server kdykoliv podle potřeby odpovědět. Po odpovědi serveru se spojení ukončí a klient ihned naváže další spojení, ve kterém může server opět odpovědět, jakmile získá potřebná data.

2.2.2 MQTT

MQTT neboli Message Queuing Telemetry Transport je otevřený síťový komunikační protokol, který slouží k přenosu zpráv mezi zařízeními určený pro rozsáhlé sítě s malým datovým tokem. Přenos zpráv přes MQTT probíhá přes TCP/IP a umožňuje zabezpečenou komunikaci pomocí kryptografického protokolu TLS.

Sítě MQTT jsou postaveny na síťové architektuře klient-server, kdy klienty mohou být různá chytrá zařízení. Server se nazývá tzv. MQTT Broker a slouží jako prostředník ke komunikaci mezi zařízeními a zároveň jako brána k připojení k internetu. Zprávy ze zařízení v síti MQTT jsou řazeny do tzv. témat, která usnadňují přístup ke konkrétním zprávám. Témata jsou hierarchicky strukturovaná a oddělená lomítky. Příkladem tématu, ve kterém teplotní senzor v místnosti č. 1 publikuje data brokeru může být:

domov/patro-1/místnost-1/teplota

Stejnou strukturu lze použít i pro čtení dat ze senzoru.

MQTT zavádí tři úrovně QoS tj. potvrzování zpráv pomocí mechanismů pro řízení datových toků.

1. **Maximálně jednou.** Zpráva je zaslána jen jednou a není potvrzována. Odesílatel nečeká na potvrzení ani na odpověď.
2. **Alespoň jednou.** Zpráva je vysílána, dokud není potvrzena příjemcem.
3. **Právě jednou.** Odesílatel i příjemce zajistí dvouúrovňové potvrzení. Tím je zaručeno doručení zpráv klientovi.

3 Rešerše dostupných systémů pro ukládání dat ze senzorů pro IoT

V současné době existuje velké množství databázových systémů, které jsou přímo určené nebo použitelné pro ukládání dat z IoT zařízení. Databázové systémy je možné rozdělit podle způsobu ukládání dat na SQL a NoSQL.

Hlavní rozdíly mezi SQL a NoSQL databázemi:

1. **Datový model databáze:** SQL databáze jsou na rozdíl od NoSQL založené na relačním datovém modelu.
2. **Schéma databáze:** SQL databáze využívají strukturovaný dotazovací jazyk a mají předdefinované databázové schéma. Databázové schéma popisuje objekty databáze a vztahy mezi nimi. NoSQL databáze mohou dynamicky měnit databázové schéma a v každém objektu mohou využívat jinou datovou strukturu.
3. **Škálovatelnost:** Ve většině případů jsou SQL databáze vertikálně škálovatelné, což znamená, že je databáze zpravidla uložena na jediném serveru a nelze je dělit na oddíly. Je-li potřeba zvýšit výkon databáze, je to možné pouze její optimalizací nebo zvýšením výkonu serveru, na kterém je databáze uložena. NoSQL databáze jsou na druhou stranu škálovatelné horizontálně. Horizontální škálování znamená, že databáze zvládne vyšší provoz rozdělením databáze na více serverů.
4. **Struktura:** SQL databáze se zpravidla skládají z tabulek, zatímco NoSQL může být tzv. key-value, dokumentová, sloupcová, grafová nebo vícemodelová.

3.1 SQL databáze

Název SQL označuje standardizovaný strukturovaný dotazovací jazyk používaný především v relačních databázích. Data v relačních databázích jsou uložena ve formě tabulek, které mezi sebou mohou být propojeny pomocí relací. Relace vytváří vztah neboli vazby mezi jednotlivými tabulkami. Podmínkou vytváření relací je to, že společné údaje obou tabulek musí být stejného typu a měly by mít stejnou délku. U relačních databází zpravidla mohou být relace trojího druhu:

- **Relace 1:1**
  Jde o nejjednodušší typ relace, která může vzniknout mezi dvěma tabulkami. Vazba 1:1 mezi tabulkami vzniká tam, kde jedna hodnota primárního klíče hlavní tabulky odpovídá právě jedné hodnotě v druhé tabulce. Tímto způsobem je možné data na jednom řádku v hlavní tabulce doplnit dalšími daty z druhé tabulky, kdy tato vazba je zaručeně 1:1. Nemůže nastat případ, kdy by primárnímu klíči z hlavní tabulky odpovídalo více řádků tabulky druhé [2].

- **Relace 1:N**

  Relace 1:N je nejpoužívanější typ spojení tabulek, kdy jednomu řádku z první tabulky odpovídá jeden, více nebo i žádný řádek druhé tabulky. Pomocí relací tohoto typu je možné propojit i více tabulek současně [2].

- **Relace M:N**

  Relace M:N značí, že každému řádku první tabulky lze přiřadit jeden, více nebo i žádný řádek ze druhé tabulky a naopak. Spojení M:N nelze vytvořit přímo, ale pomocí vytvoření dvou relací 1:N na třetí tabulku, která slouží jako převodník. Třetí tabulka sloužící jako převodník obsahuje ve dvou sloupcích hodnoty primárních klíčů z propojovaných tabulek [2].

Mezi nejčastěji používané SQL databázové systémy používané pro IoT v současné době patří:

- MySQL – databázový systém vyvíjený firmou Oracle, dostupný zdarma
- Oracle – databázový systém pro velké objemy dat
- SQLite – databázový systém s nižší výkonností, který není založený na principu klient-server
- Microsoft SQL Server
- PostgreSQL

3.2 NoSQL databáze

Termín NoSQL se často nesprávně vykládá jako označení pro Non-SQL databáze. Přesnějším výkladem názvu NoSQL je spíše Not only SQL, jelikož termín označuje volně specifikovanou třídu nerelačních databází, z nichž některé databáze podporují i jazyk SQL nebo jeho deriváty.

NoSQL databáze ukládají data jiným způsobem než relační databáze. Zpravidla se nejedná o tabulkové databáze, ale využívají různých druhů ukládání v závislosti na datovém modelu. Mezi hlavní druhy NoSQL databází patří:

- **Dokumentové databáze** ukládající data do dokumentů s formátem podobným JSON objektům. Každý dokument obsahuje dvojice proměnná – hodnota. Dalším případem dokumentové databáze může být i tzv. XML databáze, kde se data ukládají mezi XML značky.
- **Key-value store** patří mezi nevíce zjednodušené druhy databází. Data jsou vždy uložená v páru klíč-hodnota.
- **Sloupcově orientované databáze** jsou navržené pro velmi velké množství dat. Ukládají data do tabulek, ale jiným způsobem než běžné relační databáze. Název a formát každého sloupce může být pro každý řádek tabulky jiný. Sloupcově orientované databáze lze vyložit jako dvoudimenzionální databázi Key-value [20].
- **Grafové databáze** využívají struktury skládající se z vrcholů a hran a zpracovávají data v podobě grafů. Jsou vhodné například pro modelování mobilní sítě, neuronové sítě a podobně.

3.2.1 Firebase

Firebase je sada nástrojů společnosti Google pro vytváření mobilních a webových aplikací. Platforma Firebase nabízí hosting, realtime databázi, cloudové úložiště a mnoho dalších nástrojů pro zjednodušení autentifikace a přihlašování uživatelů. Součástí prostředí je NoSQL real-time databáze, která zpracovává data podobně jako JSON. Přístup k webu a databázi je možný přes Javascriptové klientské API, ale jsou podporovány i další programovací jazyky, například PHP [17] [21].

3.3 ThingSpeak

ThingSpeak je analytická služba pro internet věcí umožňující sběr, vizualizaci a analýzu datový toků v cloudu. Je to systém vyvíjený společností MathWorks Inc, která je známá především vývojem programovací a výpočetní platformy MATLAB a jeho nadstavby pro simulace a modelování dat Simulink. Data odeslaná do ThingSpeak je možné okamžitě zobrazovat a zpracovávat například i pomocí kódů pro Matlab. Platforma se často používá pro vytváření prototypů systémů v síti IoT [18].

Mezi klíčové vlastnosti této služby patří:

- Podpora populárních protokolů IoT,
- Vizualizace dat v reálném čase,
- Získávání dat ze zdrojů třetích stran,
- Možnost využití skriptů platformy Matlab,
- Automatická analýza dat,
- Propojení s Twitterem a podobnými aplikacemi.

3.4 MS Azure

Microsoft Azure je výpočetní cloudová platforma, která umožňuje vytváření a hostování webových aplikací prostřednictvím datových center společnosti Microsoft. Nabízí Azure SQL relační databázi vytvořenou pro cloud. Databáze podporuje automatizované funkce s umělou inteligencí. Výhodou je možnost využití bezserverového výpočetního prostředí a automatického škálování prostředků dle potřeby. Velikost databáze a množství prostředků jsou tak omezeny prakticky jen cenou, kterou je uživatel platformy MS Azure ochoten zaplatit. Kromě SQL databáze Azure nabízí novou NoSQL databázi Azure Cosmos DB, službu Linux Virtual Machines, Blob Storage, VPN Gateway a mnoho dalších nástrojů [19].

4 Návrh systému sběru a vizualizace dat

Tato kapitola se zabývá výběrem webhostingu a databáze a návrhem systému sběru dat ze zařízení internetu věcí. V první části této kapitoly je popis požadavků na vybíraný webhosting a databázi. Druhou část tvoří návrh systému pro sběr a zpracování dat a návrh struktury databáze.

4.1 Výběr webhostingu a databáze

Vzhledem k tomu, že firma Home Technologies je v úplném počátku svého působení, nemá k dispozici žádný svůj server ani webové stránky. Součástí zadání diplomové práce je proto i výběr vhodného systému pro správu firemních webových stránek, a především systému pro sběr, zpracování a následnou vizualizaci dat. Mezi požadavky pro výběr webhostingu patřily následující body:

1. **Dostatečně velká databáze** a možnost dokoupení kapacity v případě potřeby.
2. **Doména druhého řádu** – možnost vytvořit webovou stránku tak, aby obsahovala pouze název_firmy.cz. Některé levné a všechny neplacené webhostingy si kladou podmínku zahrnutí jejich názvu do adresy webové stránky například název_firmy.webhosting.cz.

   V současné době (březen 2021) má společnost Home Technologies registrované a/nebo rezervované následující domény:
   - hometechnologies.cz a další národní domény v rámci Evropy.
   - hometechnologies.net, hometechnologies.store (doména pro eshop) a mnoho dalších.
3. **Velikost datového úložiště pro samotný web** – s větším datovým úložištěm je možné na webové stránky nahrát více obsahu, jako třeba obrázky, animace, dokumenty, videa apod.
4. **Úroveň zabezpečení** – podpora v dnešní době stále častěji používané šifrované komunikace pomocí protokolu HTTPS.
5. **Technická podpora** – některé webhostingy poskytují technickou podporu k jejich produktům.
6. **Podporované programovací jazyky** – prvotním požadavkem byla podpora software ASP.NET, ale později společnost došla k rozhodnutí, že bude dostačující použít klasické jazyky pro tvorbu webu. Mezi tyto jazyky patří HTML, PHP, CSS a JavaScript.
7. **Neomezené množství přístupů k webu**
8. **Rychlost odezvy a garance dostupnosti webu**
9. **Cena**

Varianta s vytvořením vlastního serveru byla zavrhnuta kvůli mnoha překážkám jako jsou nákup vhodného hardware, zajištění dostatečné rychlosti připojení k internetu a zřízení veřejné IP adresy.

Tabulka 2 Seznam zvažovaných webhostingů

Název hostingu	Velikost webu (GB)	Velikost databáze (GB)	Doména 2. řádu	Cena (Kč)
Aspify.com	Neomezena	0,1–25	✔	0–450
ASPone.cz	1–30	0,1–3	✔	0–750
Google Cloud hosting	Neomezena	Neomezena[2]	✔	-[3]
Wedos.cz	Neomezena	1–2	✔	33–121
Czechia.com	10–40	2–10	✔	196-916
Forpsi.com	Neomezena	Neomezena[1]	✔	20-145
Vlastní server	Neomezena	Neomezena	✔	?

(data převzata z webů poskytovatelů webhostingu v srpnu 2019)

Původním záměrem bylo využití ASP.NET pro zdrojový kód webových stránek a zpracování dat. Software pro tvorbu webových aplikací ASP.NET lze použít na různých platformách a jde o tzv. otevřený neboli svobodný software. ASP.NET je rozšíření vývojářské platformy .NET, to znamená, že webové aplikace a služby je možné psát v kterémkoliv jazyce podporujícím Common Language Runtime, například C#, Visual Basic .NET, JScript.NET ale i v Pythonu. V roce 2019, kdy probíhal výběr webhostingu, nebylo mnoho poskytovatelů, kteří by nabízeli ASP.NET a splňovali další podmínky výběru za přijatelnou cenu. Z tohoto důvodu, a z důvodu menší základny uživatelů softwaru ASP.NET byla vybrána varianta webu s použitím PHP jako hlavního programovacího jazyku. Další nevýhodou použití jazyka ASP.NET je poměrně malá dostupnost tutoriálů a dalších návodů k psaní v tomto jazyce. Naproti tomu skriptovací jazyk PHP má výrazně větší uživatelskou základnu, a tudíž je dostupnější co se týče webhostingů a znalostní báze.

Po dlouhém zvažování výhod a nevýhod všech dostupných webhostingů byl ve spolupráci s firmou Home Technologies vybrán poskytovatel webhostingu Wedos. Výhodami Wedosu ve srovnání s konkurencí byla nižší cena a možnost rozšíření webu o tzv. VPS ON, tj. služba postavená na velkokapacitní profesionální cloudové infrastruktuře, která umožňuje libovolně rozdělovat výkon a prostředky serveru. Dalším bodem pro výběr poskytovatele byla skutečnost že Wedos je největší a nejprodávanější hosting v České republice.

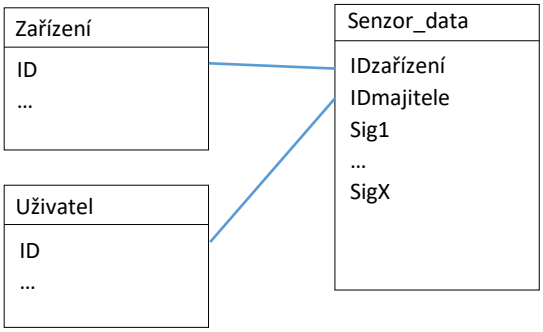
S výběrem poskytovatele webhostingu Wedos však odpadla možnost výběru databáze, jelikož Wedos nabízí k novým službám pouze databázi MariaDB.

[2] S velikostí databáze stoupá i cena
[3] Cena se liší podle využití

4.2 Návrh struktury databáze

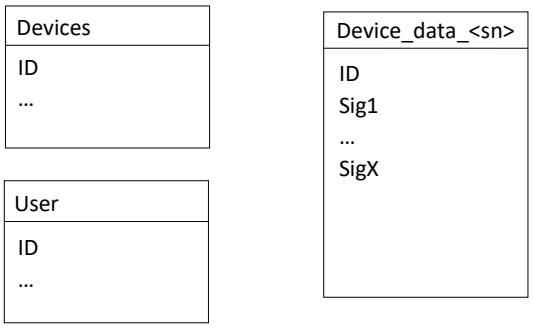
Jelikož poskytovatel hostingu Wedos nabízí pro své nově vytvořené weby pouze relační databázi MariaDB vycházející z MySQL, je vhodné navrhovat tuto databázi tak, aby co nejlépe využívala relačního modelu. Správně navrhnutá relační databáze pak umožňuje data efektivněji ukládat, třídit a prohledávat. Podstatou správného návrhu struktury dat relační databáze je rozdělení dat tak, aby se nedaly dále dělit a s tím předejít i redundanci – nadbytečného opakování stejných hodnot v databázi.

Struktura databáze musí v základu obsahovat tabulky dat z jednotlivých senzorů, tabulku obsahující seznam všech zařízení s bližšími informacemi o senzorech a tabulku se seznamem uživatelů.



Obrázek 2 První návrh základní struktury databáze

První návrh základní struktury databáze, viz Obrázek 2, se brzy projevil jako nevhodný, neboť data ze všech senzorů by se ukládala do jedné tabulky „*Senzor_data*“, která by v budoucnu obsahovala velmi velký počet řádků, což by mohlo negativně ovlivňovat rychlost databáze. Mezi další nevýhody tohoto řešení je možné zařadit počet signálů u každého řádku tabulky, jelikož senzor BME používá všech sedm signálových sloupců, kdežto senzor RLY pouze signály 1 až 3 a další by byly nevyužity. Z toho důvodu bylo rozhodnuto vytvořit pro data z každého senzoru speciální tabulku s názvem „*Device_data_<sn_zařízení>*“. Tímto způsobem ukládání dat není možné zajistit relaci mezi tabulkou dat a tabulkami uživatelů a zařízení, ale mělo by to zajistit rychlejší přístup k datům konkrétního senzoru v případě, že tabulka bude obsahovat mnoho řádků.



Obrázek 3 Upravený návrh základní struktury tabulek

Tyto tabulky budou propojeny relacemi přes jiné tabulky a dále doplněny o další tabulky k nastavení oznámení, k uložení rozložení stránky dashboard, k uchování dat pro ovládání relé akčních členů, tabulku pro správu přístupových tokenů atd.

4.2.1 Návrh tabulek s daty ze senzorů

Společnost Home Technologies od roku 2019 vyvíjí tři chytrá zařízení, ze kterých se vyvodily požadavky pro ukládání dat do databáze. Prvním z těchto zařízení je senzor teploty a vlhkosti označený jako BME, který mimo teploty a vlhkosti měří i atmosférický tlak. Tyto hodnoty jsou dále doplněny o výpočet pocitové teploty, teploty rosného bodu a orientační nadmořské výšky. Společně s údajem o velikosti napájecího napětí senzoru BME se do databáze bude odesílat celkem sedm datových signálů označených jako *sig1* až *sig7*.

Druhým zařízením společnosti Home Technologies je dvoukanálový relé akční člen označený jako RLY, který do databáze bude odesílat hodnotu o stavu prvního a druhého kanálu a velikost napájecího napětí. Třetí zařízení je senzor obsahu poštovní schránky, který má označení PBX. Senzor obsahu poštovní schránky do databáze bude posílat celkem 4 signály – stav obsahu poštovní schránky, údaj určující s jakou pravděpodobností se ve schránce nachází pošta, intenzitu okolního osvětlení a velikost napájecího napětí. Některé hodnoty ze senzorů jsou určeny především pro identifikaci možné chyby senzoru, například vlivem nízkého napájecího napětí.

Tabulka 3 Navrhované hlavičky tabulek Device_data_<sn> v databázi

Název zařízení	**Index řádku tabulky**		**Časové razítko**		**Data**	
	Název	**Datový typ**	**Název**	**Datový typ**	**Název**	**Datový typ**
BME	ID	INT	serverdatetime	timestamp	sig1 až sig7	float
RLY	ID	INT	serverdatetime	timestamp	sig1 až sig3	float
PBX	ID	INT	serverdatetime	timestamp	sig1 až sig4	float

Pro zjednodušení automatického vytváření tabulek senzorů v databázi jsou názvy sloupců tabulek navrhnuty v jednotném tvaru. Každá tabulka tedy obsahuje unikátní ID každého řádku tabulky jako primární klíč, sloupec s časovým razítkem udávající přesný čas, kdy byla hodnota změřena a různý počet sloupců s daty. Počet signálových sloupců se liší podle typu senzoru, viz Tabulka 3. Tabulka s touto strukturou je použitelná pro každý typ zařízení, protože názvy sloupců jsou univerzální. Při prvním připojení zařízení do databáze se vytvoří tabulka s názvem „*Device_data_<sn_zařízení>*“ s požadovaným počtem signálových sloupců, podle sériového čísla zařízení.

4.2.2 Návrh tabulky se seznamem připojených zařízení

Kromě tabulky s daty ze senzorů je potřeba vytvořit také tabulku obsahující bližší informace o jednotlivých zařízeních. Tato tabulka se nazývá „*Devices*“ a obsahuje následující sloupce:

- **ID** zařízení s datovým typem unsigned INT.
- **Sn** obsahující sériové číslo konkrétního senzoru. Sériové číslo je unikátní kód přiřazený k zařízení již při výrobě. Ze sériového čísla lze mimo jiné rozpoznat typ zařízení (RLY, BME, PBX apod.). Sn se skládá ze třech znaků označujících typ zařízení a dvanácti hexadecimálních znaků označujících interní informace společnosti Home Techonologies. Dohromady sériové číslo nesmí nikdy sestávat z více než patnácti znaků. Datový typ pro sloupec „Sn“ je varchar.
- **ServerDateTime** udává datum a čas, kdy byla zaznamenána poslední aktivita zařízení. Tato hodnota není získávána ze zařízení, ale do tabulky ji připojuje server. Formát časového razítka je v programovacím jazyce PHP definován jako „Y-m-d H-i-s“, v praxi obsahuje například hodnotu 2020-02-19 20:49:10. Datový typ sloupce ServerDateTime je datetime.
- **Name** obsahující jméno zařízení, které si uživatel může sám zvolit. Jelikož je v databázi nastaveno kódování UTF8, může název obsahovat i českou diakritiku. Délka políčka jména zařízení je omezena na 40 znaků datovým typem varchar.
- **SSID** je název wifi sítě, ke které je zařízení právě připojeno. Tato hodnota se ukládá v databázi pro případ řešení potíží s připojením. Maximální délka SSID je stanovena standardem IEEE 802.11 na 32 znaků, proto je stejné omezení nastaveno i v databázi datovým typem varchar. Zobrazení názvu SSID není uživatelům umožněno.
- **ConnectionType** je proměnná definována jako výčtový typ obsahující druh připojení zařízení k internetu. Může nabývat hodnot „wifi“, „lan“, „ap“. Tato hodnota slouží především pro účely řešení problémů a není dostupná uživatelům.
- **RSSI** je identifikátor síly přijímaného wifi signálu uložený do databáze jako hodnota v procentech. Opět slouží pouze pro řešení problémů a není dostupný uživatelům. RSSI má v databázi přiřazen datový typ unsigned tinyint, což odpovídá jednobytovému integeru. Podporovaný rozsah zvoleného datového typu 0-255 je dostatečný pro hodnotu udávající procenta.
- **MAC** obsahující MAC adresu zařízení i s oddělujícími dvojtečkami. MAC adresa podle IEEE 802 sestává ze šesti bytů a rozšíření její délky se neočekává dříve než v roce 2100. Délka MAC adresy v databázi je tedy omezena na 17 znaků, což je dostatečné pro zobrazení šesti bytů v hexadecimálním tvaru a pět oddělujících znaků. Datový typ pro ukládání MAC adresy je zvolen jako varchar. MAC adresa taktéž není dostupná pro uživatele.
- **Voltage** obsahuje velikost napájecího napětí v datovém typu float. Slouží pro detekování chyb způsobených nízkým napájecím napětím kvůli nevhodnému napájecímu zdroji či příliš dlouhému přívodnímu vodiči k zařízení.

- **Diagnostika** je hodnota obsahující číslo chyby detekované zařízením. Příkladem detekované chyby může být například chyba číslo 8002 (žádná odpověď od serveru v předchozím požadavku). Datový typ sloupce Diagnostika je unsigned smallint tj. dvoubytový integer, může tedy obsahovat hodnoty 0-65535.
- **Firmware** je sloupec obsahující označení pro verzi firmware aktuálně nahranou do zařízení. Ukládání této hodnoty umožňuje vyslat uživateli upozornění o neaktuální verzi firmware v jejich zařízení. Datový typ pro sloupec Firmware je nastaven jako varchar.
- **UpdateTime** je hodnota udávaná v sekundách, která označuje interval odesílání dat ze zařízení do databáze. Systém tak pozná, že se zařízení „odmlčelo“ déle, než mělo a podle této hodnoty mění barevné označení stavu zařízení. Zelená znamená online, oranžová označuje dobu neaktivity delší než čas udávaný v UpdateTime, ale menší než trojnásobek UpdateTime, a červená barva označuje zařízení, které je neaktivní delší dobu.
- **IP** obsahuje IP adresu zařízení v lokální síti. Datový typ pro IP adresu je varchar, hodnota může obsahovat až 39 znaků, jelikož se předběžně počítá s IPv6, kde IP adresa v základu obsahuje 8 skupin po čtyřech znacích + 7 oddělovacích znaků.
- **IPPublic** obsahuje veřejnou IP adresu. Tuto hodnotu zapisuje do databáze server, který při zpracovávání HTTP POST požadavku detekuje IP adresu klienta. Veřejná IP adresa se používá pro detekování, zda je počítač, ve kterém se zobrazují data, ve stejné síti jako zařízení. Pokud ano, nabídne systém přímý přístup k nastavení zařízení pomocí lokální IP adresy.
- **FirstLog** je sloupec obsahující datum a čas vytvoření prvního záznamu o zařízení v tabulce.
- **Owner** obsahující ID majitele daného zařízení. Hodnota sloupce Owner bude propojena pomocí relace s hodnotou ID konkrétního uživatele z tabulky *User*.

Hodnoty do tabulky „*Devices*“ jsou nahrávány pomocí SQL příkazu UPDATE, to znamená, že se do databáze hodnota nahraje jen při změně dat. Každé zařízení má v této tabulce jen jeden řádek s daty. Podle počtu řádků v tabulce *Devices* lze jednoduše zjistit počet všech zařízení, která kdy s databází komunikovala.

4.2.3 Návrh tabulky uživatelů

Jelikož je nutné zajistit, aby neměli všichni návštěvníci webu hometechnologies.cz přístup k datům ze senzorů, musí web obsahovat možnost přihlášení. Každý uživatel, který se zaregistruje do systému Home Technologies MARS, má automaticky vytvořen řádek se svými údaji v tabulce *User*. V tabulce *User* se ukládají následující data:

- **ID** uživatele v datovém typu INT, které se automaticky přičítá s každým dalším novým uživatelem.
- **Mail** uživatele sloužící jako uživatelské jméno. Dokument RFC 3696 [12] udává, že maximální délka emailové adresy může být až 320 znaků a všechny weby by s touto hodnotu měly počítat.
- **PSWD –** heslo uživatele zašifrované pomocí PHP funkce password_hash(), která používá algoritmus bcrypt. S postupným vývojem jazyka PHP se mění i jeho základní algoritmus pro šifrování hesel ve funkci password_hash(). Doporučená délka databázového řádku pro textový řetězec šifrovaného hesla je 255 znaků datového typu varchar.
- **Tel** – sloupec databáze obsahující telefonní číslo uživatele i s národní předvolbou ve tvaru se znakem „+“ na začátku. Podle technického doporučení ITU E.164 [14] může telefonní číslo obsahovat až patnáct číslic.
- **Name** obsahující jméno uživatele omezené na maximální délku 32 znaků. Pro sloupec Name je nastaven datový typ varchar s omezením na 32 znaků.
- **MidName** – sloupec, kde je možné vyplnit a uložit druhé jméno, či jiné označení uživatele. Maximální délka textového řetězce je zde nastavena taktéž na 32 znaků s datovým typem varchar.
- **SurName** – sloupec pro uložení příjmení uživatele, omezený na maximální délku příjmení 32 znaků. Datový typ sloupce SurName je varchar.
- **UserState –** slouží pro rozlišení práv uživatele, kdy políčko UserState každého uživatele obsahuje číslici, pomocí které web rozlišuje, jaký obsah se uživateli zobrazí a co může daný uživatel měnit. Hodnota je ukládána s datovým typem unsigned tinyint, může tak nabývat hodnot 0-255. V současné době hodnota UserState rozlišuje pouze tyto druhy uživatelů:
    - **1** – Běžný uživatel – Každý nový účet je založen s právy pouze pro zobrazení základního obsahu webu.
    - **2** – Admin – Administrátorské účty mají přístup k veškerému obsahu webu.
    - **3** – Editor – uživatel typu editor může editovat některé součásti webu.
    - **4** – Překladatel
- **RegDate –** jak již napovídá anglický název, sloupec se jménem RegDate obsahuje datum a čas registrace uživatele. Web pomocí této hodnoty může rozlišovat nové uživatele a podobně. Datový typ hodnoty RegDate je timestamp.
- **City –** obsahující místo bydliště uživatele s maximální délkou textového řetězce omezenou na 28 znaků s datovým typem varchar.

- **Street** – ulice bydliště uživatele. Datový typ hodnoty je varchar s maximální délkou omezenou na 30 znaků.
- **CP** – pro uložení čísla popisného a orientačního s datovým typem varchar a maximální délkou omezenou na 10 znaků.
- **Country** – obsahující kód státu, ve kterém uživatel žije. Hodnota je datového typu varchar.
- **ZIP** – obsahující poštovní směrovací číslo s datovým typem varchar.
- **PrefLang** – obsahující kód preferovaného jazyka uživatele, ve kterém se mu bude zobrazovat webové rozhraní společnosti Home Technologies.

4.3 Odhad velikosti databáze

Jelikož se do databáze bude ukládat poměrně velké množství dat, je vhodné provést odhad velikosti databáze. Dle výsledků odhadu předpokládané velikosti uložených dat v databázi bude možné předvídat, jak dlouho bude trvat, než se databáze zaplní. Současně se také rozhodne, jestli bude nutné dokoupit velikost úložiště databáze, anebo postačí odebírat staré záznamy.

Největší nároky na velikost databáze budou mít tabulky obsahující data ze senzorů, například senzor teploty BME odesílá nová data každých 300 sekund. V případě senzoru teploty a vlhkosti BME se odesílá do databáze celkem sedm datových signálů datového typu float současně s časovým razítkem přijetí dat. Datový typ float databáze MariaDB má velikost 4 byty, datový typ timestamp zabírá 4 byty a index tabulky typu INT 4 byty pro data a 4 byty pro primární klíč [13]. Datová tabulka senzoru teploty vlhkosti obsahuje 9 sloupců hodnot po 4 bytech + 4 byty pro primární klíč, což odpovídá celkem 40 bytům na každý řádek. Tabulka s daty ze senzoru obsahu poštovní schránky obsahuje jen 4 sloupce s daty a tabulka dat z relé akčního členu obsahuje pouze 3 sloupce dat, viz Tabulka 4.

Tabulka 4 Hodnoty pro výpočet velikosti jednoho řádku datové tabulky

Zařízení	Označení sloupce tabulky a velikost jedné hodnoty [B]									Klíč	Celkem
	ID	serverdatetime	sig1	sig2	sig3	sig4	sig5	sig6	sig7	[B]	[B]
BME	4	4	4	4	4	4	4	4	4	4	40
RLY	4	4	4	4	4	-	-	-	-	4	24
PBX	4	4	4	4	4	4	-	-	-	4	28

Při opakování požadavku každých 300 sekund provede jeden senzor BME 288 požadavků každý den. Denně by tak tabulka spotřebovala minimálně 11520 bytů, za rok více než 4,2 MB.

Tabulka 5 Odhad minimální velikosti datové tabulky jednoho zařízení BME

Předpokládaná minimální velikost jednoho řádku zařízení BME	40 B
Předpoklad velikosti za den, při opakování požadavku po 300 s	11 520 B
Předpokládaná velikost datové tabulky za měsíc	357 120 B
Předpokládaná velikost datové tabulky za rok	4 204 800 B

Předpokládané velikosti datových tabulek obsahují pouze velikosti dat a nepočítají s využitím dalších prostředků úložiště, které jsou nutné pro funkci databáze. U všech předešlých výpočtů je nutné ještě přičíst velikosti dat sloužících k vytvoření tabulky, umístění tabulky do struktury databáze a další nezanedbatelná data, která nepochybně velikost tabulek ještě podstatně navýší.

Další tabulky v databázi se již nebudou vyskytovat v takovém množství a v takovém rozsahu, a proto se neočekává výrazný dopad na spotřebu prostředků úložiště. Například tabulka *Devices* obsahující seznam všech zařízení připojených k databázi a další informace o zařízeních by při tisíci připojených zařízení obsahovala „jen“ tisíc řádků se záznamy.

Tabulka 6 Odhad velikosti jednoho řádku tabulky Devices

Název sloupce	**Datový typ**	**Délka**	**Velikost [B]**
ID	int	9	4
Sn	varchar	15	16
ServerDateTime	datetime	-	8
Name	varchar	40	41
SSID	varchar	32	33
ConnectionType	enum	-	1
RSSI	tinyint	5	1
MAC	varchar	17	18
Voltage	float	-	4
Diagnostika	smallint	4	2
Firmware	varchar	9	10
UpdateTime	smallint	6	2
IP	varchar	39	40
IPPublic	varchar	39	40
FirstLog	timestamp	-	4
Owner	mediumint	6	3
		Celkem:	**227**

Velikost jednoho řádku je v případě tabulky *Devices* podstatně větší, jelikož tabulka obsahuje především textové řetězce. Předpokládaná velikost jednoho řádku tabulky *Devices* je 227 bytů. Odhad vznikl součtem velikostí datových typů s daty o maximální nastavené délce. V případě číselných datových typů se velikost buňky neliší podle přednastavené délky, ale zůstává stejná podle velikostí daných datových typů. U textových datových typů se velikost potřebné paměti liší v závislosti na délce řetězce. Velikost datového typu varchar se podle dokumentace databáze MariaDB vypočte jako součet délky konkrétního řetězce + 1 byt [13].

5 Realizace systému sběru a vizualizace dat

Společnost Home Technologies ve své nabídce chytrých zařízení nabízí například senzor atmosférických hodnot, který je schopen měřit teplotu, vlhkost a barometrický tlak. Toto zařízení se zabudovaným jednodeskovým počítačem a wifi modulem umožňuje připojení k síti wifi, přes kterou odesílá data na web hometechnologies.cz, kde se ukládají do databáze a dále zpracovávají. Jelikož SQL databáze od poskytovatele webhostingu Wedos neumožňuje přístup k databázi odjinud než z webových stránek, ke kterým je navázána, musí všechna data procházet přes webové rozhraní. V případě webu *hometechnologies.cz* se webové rozhraní pro komunikaci se senzory nachází na adrese *./db/in.php*.

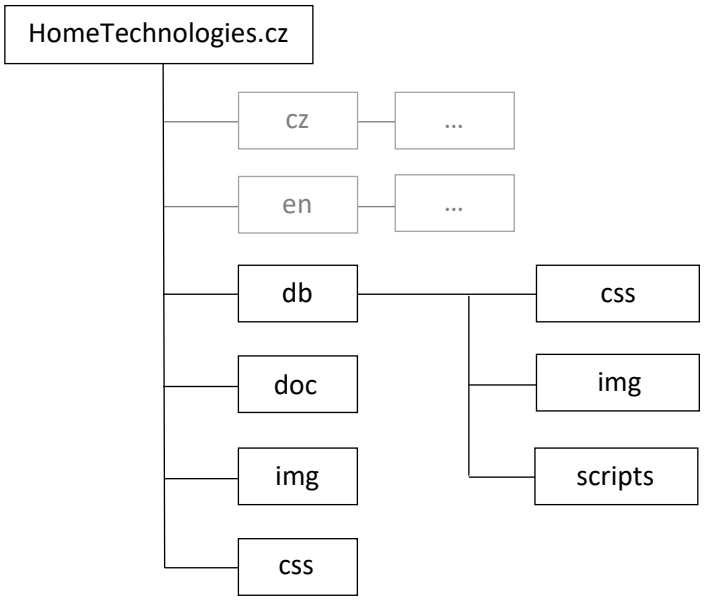
5.1 Vytvoření a správa webu

Jak již bylo popsáno v předešlých kapitolách, pro realizaci systému sběru a vizualizace dat byl vybrán webhosting od společnosti Wedos. Wedos ke svému hostingu nabízí i registraci nových domén anebo správu stávající domény. V tomto případě se spolu s webhostingem registrovala i nová doména www.hometechnologies.cz.

Po vytvoření webhostingu a propojení s doménou, Wedos na registrační email odesílá přístupová hesla pro připojení na FTP server a pro přístup k databázi. Jako FTP klient pro přenos souborů byl zvolen program Total Commander, který umožňuje přímý přístup do adresářové struktury webu. Jednotlivé webové stránky, skripty či soubory s kaskádovými styly jsou poté upravovány v editoru Visual Studio Code.

5.1.1 Struktura webu

Webová stránka na adrese www.hometechnologies.cz není využívána pouze pro potřeby systému pro sběr a vizualizaci dat, ale také jako web společnosti Home Technologies s.r.o. Z toho důvodu je web rozdělen na více podsložek, viz Obrázek 4.



Obrázek 4 Struktura složek webu HomeTechnologies.cz

Podsložky označené národními kódy (cz, en) jsou určeny pro různé jazykové mutace hlavního webu společnosti Home Technologies. Po zadání adresy www.hometechnologies.cz do prohlížeče web rozpozná jazyk prohlížeče uživatele a automaticky jej přesměruje na příslušnou podstránku. Část webu se stránkami společnosti v různých jazycích není součástí diplomové práce.

Podsložka „doc“ slouží jako úložiště pro uživatelské příručky, certifikáty, firmware a další dokumenty k jednotlivým zařízením společnosti. V podsložce „img“ jsou umístěny obrázky a animace, ke kterým se přistupuje z hlavní stránky a současně i ze systému pro sběr dat. Podsložka „css“ obsahuje kaskádové styly společně využívané v obou částech webu.

Systém pro sběr a vizualizaci dat se nachází v podsložce označené jako „db“ od slova databáze, přestože databázi přímo neobsahuje. Podsložka „db“ má vlastní strukturu s podsložkami pro soubory a obrázky používané pouze v této části stránky. Přímo ve složce „db“ jsou uloženy všechny stránky a skripty použité v systému.

5.1.2 HTTPS

Jelikož poskytovatel webhostingu Wedos nabízí v základní nabídce pouze nezabezpečený internetový protokol HTTP, bylo nutné dokoupit podporu šifrovaného zabezpečeného protokolu HTTPS. V tomto případě byla vybrána varianta HTTPS s certifikátem Let's Encrypt. Pro podporu protokolu HTTPS je nutné zajistit, aby všechny odkazy na webu vedly pouze na zabezpečené zdroje. V případě, že by web obsahoval nezabezpečené odkazy, nebylo by možné získat certifikát pro zabezpečený přístup.

Dalším důležitým krokem k implementaci zabezpečeného protokolu HTTPS na web je úprava konfiguračního souboru „*.htaccess*“. V tomto konfiguračním souboru se nastavují pravidla přesměrování webu například na chybové stránky (404 apod), dále umožňuje zaheslování přístupu k adresáři nebo souboru, přepisování URL a vytváření subdomén. Pro správnou funkci protokolu HTTPS byla nastavena následující pravidla viz Obrázek 5.

```
# Nepresmerovavat in.php
RewriteCond %{HTTPS} ^off$
RewriteCond %{REQUEST_URI} !db/in\.php$
RewriteRule ^.*$ https://%{SERVER_NAME}%{REQUEST_URI} [L,R=301]

# Presmerovani pri chybe
ErrorDocument 404 /db/404.php
```

Obrázek 5 Úryvek konfiguračního souboru .htaccess pro nastavení podpory HTTPS

Pomocí úryvku kódu na obrázku 5 je zajištěno automatické přesměrování na adresu začínající textem *„https://“* pro všechny stránky vyjma db/in.php. Tato výjimka umožňuje komunikaci se systémem i zařízením, které nepodporují HTTPS, současně ale zachovává podporu HTTPS pro zařízení se zabezpečeným připojením. Ve spodní části úryvku kódu je příklad přesměrování na stránku */db/404.php* v případě chyby 404 – Požadovaný soubor nebyl nalezen.

5.1.3 Vytvoření a správa databáze

Pro přístup a správu databáze je použito webové rozhraní phpMyAdmin, pomocí kterého byla vytvořena celá základní struktura databáze podle návrhu z kapitoly 4.2. Dále jsou pomocí jmenovaného uživatelského prostředí vytvářeny relace mezi základními tabulkami. V rozhraní phpMyAdmin je také prováděna pravidelná záloha databáze.

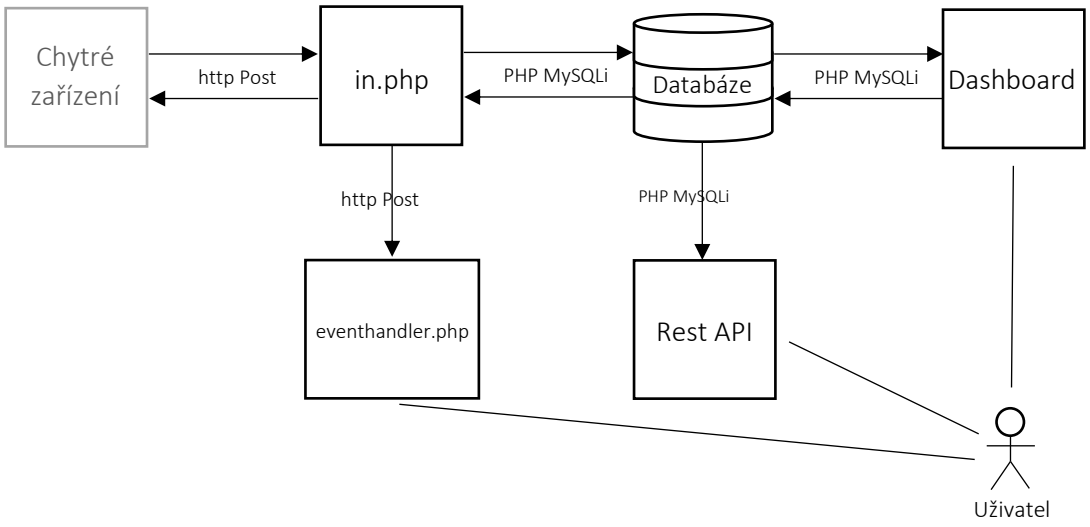
5.2 Základní struktura systému sběru dat

Ideou pro vytvoření systému sběru dat je vytvoření webového rozhraní, na které budou chytrá zařízení posílat své požadavky a data. Ačkoliv vývoj chytrých zařízení není součástí diplomové práce, pro zajištění komunikace mezi webem a chytrým zařízením je nutné definovat komunikační protokol a určit data, která se budou přenášet. Nejjednodušším druhem přenosu dat mezi chytrým zařízením a webovou aplikací je komunikace pomocí http protokolu a metody GET, kdy jsou data ze zařízení přímo přiložena do URL adresy dotazovaného webu.

Metoda GET pro příjem dat se postupem času ukázala jako nevhodná, protože odesílaná data jsou součástí URL adresy, a tudíž nejsou nijak chráněna před přečtením a přepsáním. Současně jsou hodnoty ukládány do cache paměti a při opakovaných požadavcích s malými změnami by mohly být tyto změny ztraceny. Vhodnější variantou pro předávání dat je použití dotazovací metody HTTP POST. V případě HTTP POST jsou hodnoty přenášeny uvnitř dotazu a nejsou viditelné v URL.

Data z jednotlivých zařízení se odesílají metodou HTTP POST do vstupní stránky in.php, odkud jsou odesílány do databáze pomocí funkce mysqli. Dále se data přeposílají pomocí metody http POST na stránku *eventhandler.php*, kde se kontroluje, jestli pro přijatá data není nastavené oznámení. V případě, že hodnota dosáhne přednastavené úrovně, odešle se uživateli oznámení pomocí emailu nebo Firebase Cloud Messaging.

Data uložená v databázi je poté možné zobrazovat na stránce *dashboard.php*, či k nim přistupovat přes REST API na adrese api.hometechnologies.cz.



Obrázek 6 Zjednodušené blokové schéma struktury systému

5.3 Registrace a přihlášení k systému

Jelikož by uživatelská data z chytrých zařízení mohla obsahovat soukromé informace, bylo rozhodnuto, že pro přístup do databázového systému bude požadováno přihlášení uživatele. Každý uživatel si musí vytvořit uživatelský účet a až poté si ke svému účtu připojí chytrá zařízení, která se identifikují pomocí unikátního sériového čísla.

5.3.1 Registrace

Pro vytvoření uživatelského účtu je obvykle vyžadováno zadání uživatelského jména a hesla pomocí webového formuláře. Zpočátku tedy postačí vytvoření jednoduchého webového formuláře napsaného v jazyku HTML v následujícím tvaru.

```
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>" method="post">
        <h2 class="ht-text-color-main">Sign Up</h2>
        <p>Please fill this form to create an account.</p>
        <label>Jméno<span class="w3-text-red">*</span></label><br>
        <input type="text" name="jmeno" class="w3-input ht-input w3-round" required>
        <label>Heslo<span class="w3-text-red">*</span></label><br>
        <input type="password" name="heslo" class="w3-input ht-input w3-round" required>
        <button type="submit">Registrovat</button>
</form>
```

Předešlý úryvek kódu obsahuje vytvoření HTML formuláře pro zadání uživatelského jména a hesla, a také nastavení cílové stránky, kam se odešle POST požadavek po kliknutí na tlačítko *Registrovat*. Po vyplnění údajů a stisknutí tlačítka registrovat vyšle stránka POST požadavek sama na sebe. V jiné části stránky požadavek zpracuje a uloží do databáze. Před uložením se ještě musí ošetřit vložené hodnoty, zda splňují podmínky pro vložení do databáze a nejsou nebezpečné. Příkladem nebezpečné hodnoty může být napadení databáze metodou SQL injection.

Před uložením do databáze musí být testováno hned několik různých situací. Jelikož je zpracování POST požadavku umístěno na stejné stránce jako jeho vytváření, musí skript rozpoznat, kdy má požadavek zpracovávat. Zpracování POST požadavku probíhá pouze v případě, že stránka přijme POST požadavek.

```
if($_SERVER["REQUEST_METHOD"] == "POST"){
…
}
```

Pomocí předchozího úryvku kódu se rozpozná, jestli se stránka spustila požadavkem POST, dále se ošetří, zda jsou hodnoty validní. Například ověření, zda je vyplněno uživatelské jméno:

```
if(empty(test_input($_POST["jmeno"]))){
  $login_err "Prosím vložte uživatelské jméno.";
}else{
  $login = test_input($_POST["jmeno"]);
}
```

Postupem času se ukázalo, že by bylo lepší využít email uživatele namísto uživatelského jména, proto byla provedena změna zpracování registrace. Předchozí úryvek byl upraven ke zpracování emailové adresy jako jména uživatele. Emailová adresa se nyní používá i jako identifikátor uživatele a součást názvu uživatelské tabulky.

V dalším kroku probíhá ošetření validity vloženého hesla, kde se kontroluje, jestli je heslo vyplněné a jestli obsahuje dostatečný počet znaků. V případě, že některá z podmínek není splněna, je o tom uživatel informován.

```
if(empty(test_input($_POST["password"]))){
      $password_err = "Please enter a password.";
} elseif(strlen(test_input($_POST["password"])) < 6){
      $password_err = "Password must have at least 6 characters.";
} else{
      $password = test_input($_POST["password"]);
}
```

Všechny hodnoty přijaté pomocí funkce POST jsou navíc ošetřeny následující funkcí, ve které se z každé hodnoty odeberou prázdné znaky jako jsou mezery, znak tabulátoru, označení konců řádků a podobně. Dále se odstraňuje znak zpětného lomítka a další znaky se nahrazují jejich vhodnými variantami.

```
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
  }
```

Jsou-li všechny hodnoty zadány správně, dojde k jejich odeslání do tabulky se seznamem všech uživatelů. Pro odeslání hodnoty do tabulky *User* je využito funkcí *mysqli*, které navíc ošetřují datové typy hodnot. V tomto úryvku je již počítáno s emailem jako uživatelským jménem. Jelikož emailová adresa obsahuje i znaky nevhodné pro použití jako název databázové tabulky, musela být adresa upravena nahrazením znaků tečky a zavináče na podtržítka.

```
$sql = "INSERT INTO User (pass, login) VALUES (?, ?)";

if($stmt = mysqli_prepare($link, $sql)){
     mysqli_stmt_bind_param($stmt, "ss", $param_password, $login);
     $param_password = password_hash($password, PASSWORD_DEFAULT);
     $par_email = $login;
     $par_email = str_replace(".","_",$par_email);
     $par_email = str_replace("@","_",$par_email);

     if(mysqli_stmt_execute($stmt)){
        $sqlcreateuser = "CREATE TABLE User_" . $par_email . " (
             id INT(9) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
             `sn_id` INT(9) NOT NULL,
             UNIQUE (sn_id),
             FOREIGN KEY (sn_id) REFERENCES Devices(id) ON DELETE CASCADE
        )";
     }
}
```

Předchozí úryvek kódu zajišťuje uložení hesla a emailové adresy jako uživatelského jména do databáze. Heslo je pomocí PHP funkce *password_hash()* zašifrováno algoritmem bcrypt a v databázi se tak ukládá pouze jeho otisk, který člověk nepřečte. V dalším kroku se upraví formát vloženého emailu a výsledná hodnota se použije jako součást názvu nově vytvořené tabulky, ve které se přiřazují zařízení uživatelským účtům. Nová tabulka obsahuje sloupec s ID zařízení, které slouží jako tzv. cizí klíč, přes který je pomocí relace spojena s tabulkou *Devices*.

Výsledný registrační formulář a uložená data v tabulce neobsahují pouze email a heslo, ale jsou doplněny o další informace o uživateli. Mezi další informace o uživateli patří například kontaktní údaje a adresa uživatele. Všechny další hodnoty jsou validovány podobným způsobem jako uživatelské jméno a heslo. Dále byl do formuláře přidán textový vstup pro opakované zadání hesla pro kontrolu. Ve skriptu je poté porovnáváno, jestli je heslo totožné s kontrolním heslem.

5.3.2 Přihlášení

Přihlášení do systému probíhá na stránce *login.php* a využívá hodnot uložených do tabulky *User* v databázi při registraci uživatele. Přihlášení se používá pro identifikaci uživatele, kterému tak server může nabízet specifický obsah dostupný pouze jemu. Jelikož je internetový protokol HTTP takzvaně bezestavový protokol a neumí uchovávat údaje o komunikaci, musí se pro rozpoznání uživatele využít jiných metod. Metoda, která se tímto problémem zabývá, se v jazyku PHP nazývá *sessions*. Sessions je možné přeložit do češtiny jako „zasedání“ anebo „relace“. Aby se odlišily označení pro *relace databáze* a *PHP relace*, bude v práci ponecháno anglické označení *PHP sessions*.

Principem přihlášení do systému pro sběr a vizualizaci dat je zobrazení webové stránky s formulářem pro vyplnění uživatelského jména – v tomto případě emailu a hesla. Po odeslání vyplněného formuláře se data kontrolují, zda neobsahují prázdné hodnoty a jsou ošetřena proti napadení databáze metodou SQL injection. Ošetření spočívá v převedení potenciálně nebezpečných znaků na html entity a odebrání zpětných lomítek a nadbytečných mezer využitím stejné funkce jako v případě registrace. Dalším krokem při přihlášení je porovnání zadaných hodnot s hodnotami uloženými v databázi pomocí PHP funkcí *mysqli*, kdy se pomocí příkazu:

```
$sql = "SELECT `ID`, `Mail`, `PSWD`, `UserState`, `Name`, `SurName` FROM User WHERE Mail = ?";
```

vysílá požadavek na získání hodnot *ID, Mail, PSWD, UserState, Name* a *SurName* z tabulky *User*, kde *Mail* odpovídá hodnotě zadané uživatelem.

Pokud je počet odpovídajících hodnot roven jedné, tzn. zadaný email se nachází v tabulce právě jednou, dojde k ověření hesla pomocí funkce *password_verify()*, kde se porovnává zadané heslo s heslem z tabulky *User* a následně se spustí PHP session funkcí *session_start()*. Po spuštění PHP session se informace o uživateli ukládají do globální proměnné $_SESSION, která je uložena na serveru.

```
if(password_verify($password, $hashed_password)){
        session_start();
        $_SESSION["loggedin"] = true;
        $_SESSION["id"] = $id;
        $_SESSION["login"] = $login;
        $_SESSION["uzivstatus"] = $user_status;
        $_SESSION["user_jmeno"] = $user_jmeno;
        $_SESSION["user_prijmeni"] = $user_prijmeni;
        redirect("/".$redir);
} else {
        $password_err = "The password you entered was not valid.";
}
```

Každá stránka, na které je vyžadováno přihlášení, musí na svém počátku před HTML tagy obsahovat volání funkce *session_start(),* což stránce zpřístupní globální proměnnou $_SESSION s uloženými informacemi o relaci uživatele. Na stránce poté stačí ošetřit citlivé hodnoty podmínkou, která porovná, jestli je proměnná *$_SESSION["loggedin"]* rovna „true“.

Pro odhlášení ze systému pro sběr a vizualizaci dat postačí pouze zrušit relaci PHP session a smazat proměnné uložené na serveru. O tyto kroky se stará webová stránka *logout.php* s následujícím obsahem:

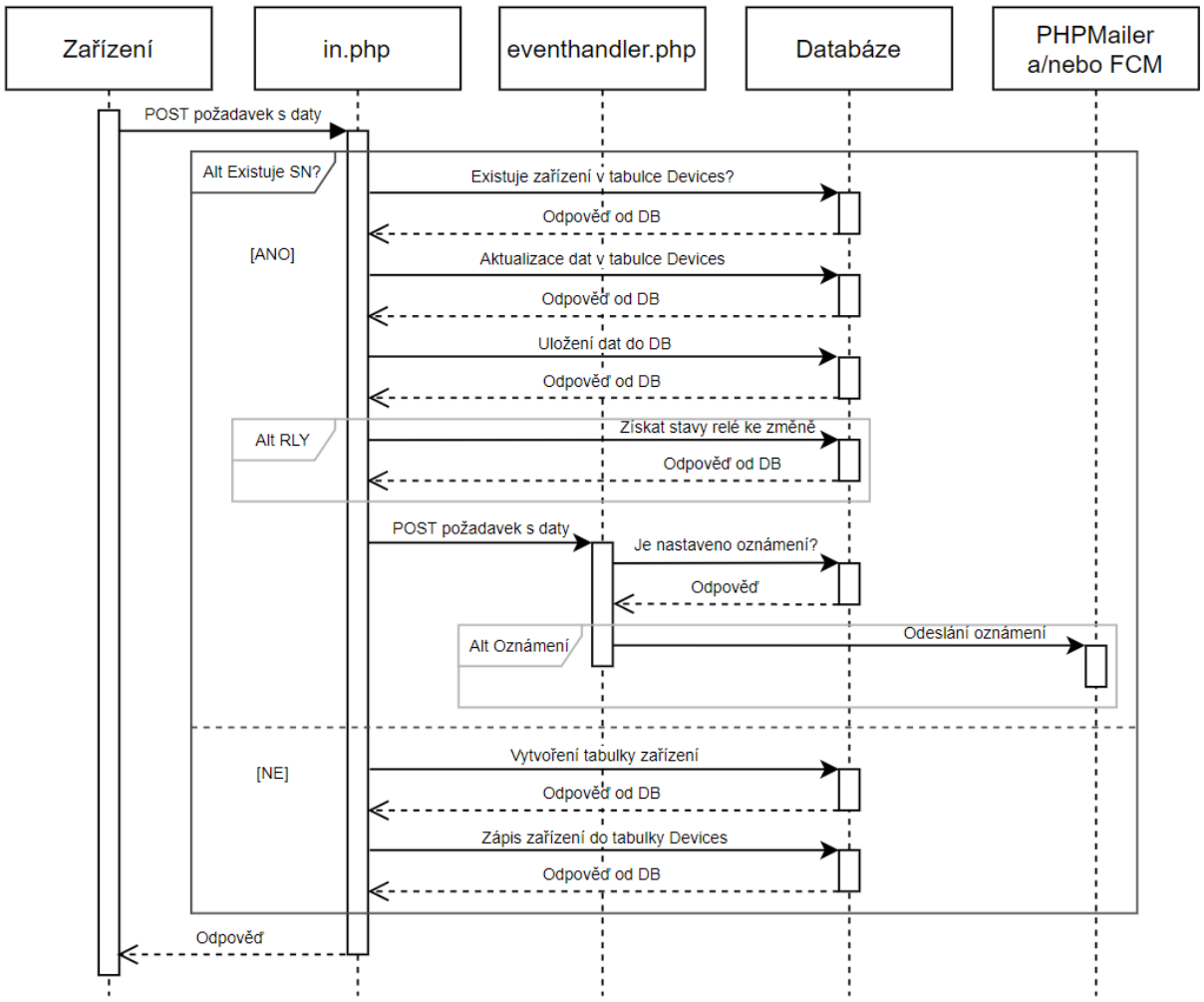
```
session_start();
$_SESSION = array();
session_destroy();
header("location: login.php");
exit;
```

Po odhlášení ze systému ukončením session a vyprázdněním globálního pole *$_SESSION* dojde k přesměrování na stránku *login.php*.

5.4 Příjem a zpracování dat

Každé chytré zařízení společnosti Home Technologies zaznamenává různé informace počínaje získáváním dat ze senzorů až po vytváření diagnostických dat o jejich funkci. Všechna takto získaná data se odesílají na stránku in.php, která je zpracovává a ukládá do databáze. Tato kapitola popisuje vývoj a funkci stránky in.php.

Stránka in.php je napsána ve skriptovacím jazyce PHP, její základ tvoří především zpracování přijatých HTTP POST požadavků. Všechny požadavky pro komunikaci jsou vyvolávány pouze z chytrých zařízení na server, jelikož v opačném pořadí – ze serveru na zařízení, není možné přes protokol HTTP komunikaci započít. Pro vytvoření požadavku je totiž nutné znát přesnou adresu dotazovaného zařízení a tu mají pouze zařízení s vlastní veřejnou IP adresou. Pro zajištění konzistence přijímaných dat se musí HTTP POST požadavky pravidelně opakovat. Intervaly opakování požadavků se liší podle typu zařízení.



Obrázek 7 UML sekvenční diagram zpracování požadavku systémem

Obrázek 7 popisuje průběh zpracování dat. Každý HTTP POST požadavek je iniciován chytrým zařízením, při kterém zařízení současně odesílá následující hodnoty:

- **Sériové číslo** zařízení sloužící jako unikátní identifikátor zařízení a typu zařízení,
- **Jméno zařízení,** které si může uživatel libovolně měnit,
- **SSID** sítě, ke které je zařízení připojeno,
- **MAC** adresu zařízení,
- **Napětí** napájecího zdroje zařízení,
- **Diagnostický kód** pro identifikaci stavu zařízení,
- **Sílu signálu**, v případě wifi sítě,
- **Lokální IP adresu zařízení,**
- **Verzi nainstalovaného firmware** v zařízení,
- **Interval odesílání požadavků**, hodnota v sekundách,
- **Typ připojení** zařízení k síti**,**
- **Datové signály,** jejichž počet se odvíjí podle typu zařízení.

Každá z výše jmenovaných hodnot má svou metaproměnnou v požadavku HTTP POST, pomocí které ji lze vyčíst z PHP pole *$_POST*. Plnění pole *$_POST,* popřípadě *$_GET* je základní funkcionalita jazyku PHP, není nutné ji inicializovat. Pro použití metod GET a POST je pouze nutné zkontrolovat, zda jsou povolené od poskytovatele, například na stránce *info.php*, nebo v nastavení PHP. V případě práce jsou metody GET a POST protokolu HTTP přednastaveny jako povolené.

Při přijetí požadavku webovou stránkou *in.php* se nejprve kontroluje, zda je vyplněn prvek pole *$_POST['sn']* obsahující sériové číslo zařízení. V případě, že není vyplněn, odpovídá stránka *in.php* kódem 501, který má stejný význam jako HTTP stavový kód „501 Not Implemented“. Je-li přijaté sériové číslo vyplněné, odesílá stránka dotaz na tabulku *Devices* v následujícím tvaru:

```
$sql = "SELECT ID FROM Devices WHERE Sn = ?";

if($stmt = mysqli_prepare($link, $sql)){
   mysqli_stmt_bind_param($stmt, "s", $sn);
   if(mysqli_stmt_execute($stmt)){
      mysqli_stmt_store_result($stmt);
      if(mysqli_stmt_num_rows($stmt) > 0){
         mysqli_stmt_bind_result($stmt, $id_in_devices);
         // SN je v tabulce Devices
         // Update tabulky Devices
         // Vložení řádku do tabulky Device_data_<sn_zařízení>
         // Získání nových stavů relé z DB
      } elseif (mysqli_stmt_num_rows($stmt) === 0){
         // SN není tabulce Devices
         // Vložení řádku se SN do tabulky Devices
         // Vytvoření tabulky Device_data_<sn_zařízení>
      }
   } else {
      // http kód 500
   }
}
```

Předchozí úryvek kódu obsahuje PHP *mysqli* dotaz na získání řádků z tabulky *Devices*, kde hodnota „Sn“ odpovídá přijatému sériovému číslu.

5.4.1 Princip zápisu hodnot do databáze

V případě, že počet výskytů sériového čísla v databázi je větší než 0, znamená to, že zadané sériové číslo je již registrované a jsou pro něj vytvořené potřebné zápisy a tabulka v databázi. Poté proběhne aktualizace dat v tabulce *Devices* pomocí příkazu:

```
$stmt = $link->prepare("
   UPDATE Devices SET Sn = ?, ServerDateTime = ?, Name = ?, SSID = ?, MAC=?, Voltage=?,
   Diagnostika=?, RSSI=?, IP=?, IPPublic=?, Firmware=?, UpdateTime=?, ConnectionType=?
   WHERE Sn = ?
");
$stmt->bind_param("sssssdiisssiss", $_POST['Sn'], date("Y-m-d H-i-s"), $_POST['Name'],
   $_POST['SSID'], $_POST['MAC'], $_POST['Voltage'], $_POST['Diagnostika'], $_POST['RSSI'],
   $_POST['IP'], $pub_ip,  $_POST['Firmware'], $_POST['UPDT'], $_POST['ConType'], $_POST['Sn']
);
$stmt->execute();
```

Kde funkce *prepare()* obsahuje SQL UPDATE požadavek na tabulku *Devices* s názvy všech sloupců tabulky k aktualizování hodnotami připojenými funkcí *bind_param(),* která obsahuje tytéž hodnoty z POST požadavku. Po aktualizaci dat v tabulce *Devices* se pomocí stejných funkcí *prepare()* a *bind_param()* vkládá nový řádek do tabulky *Device_data_<sn_zařízení>*. Rozdílná jsou zde pouze data a SQL příkaz INSERT INTO.

5.4.2 Automatické vytváření tabulek

V případě, že při POST požadavku neodpovídá přijaté sériové číslo žádné hodnotě v tabulce *Devices*, dojde k automatickému vytvoření datové tabulky *Device_data_<sn_zařízení>* pro dané sériové číslo a zápisu do tabulky *Devices.* Vložení nového řádku do tabulky *Devices* je zapsáno prakticky ve stejném tvaru jako aktualizace hodnot, popsaná v předchozí kapitole 5.4.1.

Vytvářené datové tabulky se liší podle typu senzoru. Společnost Home Technologies v současné době (duben 2021) nabízí tři zařízení, která ukládají data do databáze. Každé z těchto zařízení využívá jiný počet sloupců tabulky. Pro zjednodušení práce s tabulkami autor této práce rozhodl, že základ všech tří typů datových tabulek společně s jejich názvy bude totožný. Rozdílný počet datových signálů jednotlivých zařízení tak zasáhne pouze počet sloupců tabulky. Názvy těchto sloupců jsou pak voleny univerzální pro všechny typy tabulek. S volbou takovéto struktury datových tabulek se může systém v budoucnu libovolně rozšiřovat o různé druhy zařízení s různými počty signálů.

Při rozpoznaní nového zařízení, které ještě není zapsáno v databázi, systém ze sériového čísla identifikuje typ zařízení, podle kterého vytvoří novou datovou tabulku *Device_data_<sn_zařízení>* s předdefinovaným počtem signálů. Princip rozpoznání typu zařízení, reálných názvů jednotlivých signálů a dalších hodnot je popsán v kapitole 5.8. O vytváření nových datových tabulek se stará následující část kódu.

```
if (DeviceConfigIn($sn)==4){
   $sqlcreate = "CREATE TABLE Device_data_" . $sn . " (
      id INT(9) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
      serverdatetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
      sig1 FLOAT(8),
      sig2 FLOAT(8),
      sig3 FLOAT(8),
      sig4 FLOAT(8)
      )";
} elseif (DeviceConfigIn($sn)==7){
   …
}
```

Úryvek kódu zobrazuje princip rozpoznání velikosti tabulky podle sériového čísla zařízení, využitím funkce *DeviceConfigIn()* blíže popsané v kapitole 5.8. Zobrazený úryvek kódu slouží k vytvoření tabulky se čtyřmi datovými signály. V dalším kroku kódu se již vytváří SQL příkaz na vytvoření nové datové tabulky s názvem obsahujícím sériové číslo zařízení pro lepší identifikaci tabulky. SQL příkaz obsahuje definice jednotlivých sloupců velikostmi a bližšími parametry a předdefinovaný počet datových signálů. Po sestavení SQL příkazu dochází k jeho spuštění a ošetření výsledku.

5.4.3 Stavové kódy HTTP

Webová stránka *in.php* je nastavena tak, aby na každý požadavek odpovídala specifickou hodnotou. Při definici těchto hodnot byly vybrány veřejně známé stavové kódy HTTP. Ze stavových kódů HTTP byly převzaty pouze číselné hodnoty s jejich slovním popisem. V případě systému pro sběr a vizualizaci dat se stavové kódy neodesílají jako součást hlavičky odpovědi serveru, ale pouze jako číselná hodnota.

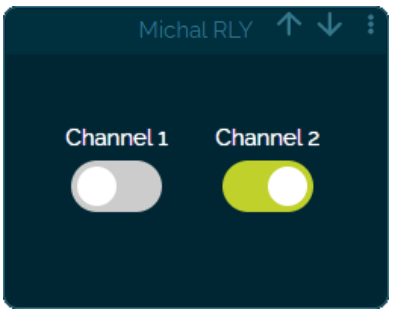
Například, zadá-li uživatel webovou stránku https://www.hometechnologies.cz/db/in.php do adresní řádky prohlížeče, zobrazí se mu pouze hodnota „501“. Prozatím jsou na stránce in.php použity následující stavové kódy:

- **200** OK
- **500** Internal Server Error
- **501** Not Implemented

5.4.4 Ovládání relé

Jelikož je mezi chytrými zařízeními společnosti Home Technologies i relé akční člen, bylo nutné zajistit komunikaci ve směru ze serveru na relé. Na rozdíl od ostatních zařízení, které pouze odesílají data na server a zpracovávají stavový kód webu *in.php* jako odpověď, musí relé akční člen s odpovědí přijmout i ovládací signály. Jelikož komunikace mezi zařízením a serverem probíhá pouze pomocí cyklicky se opakujících HTTP POST požadavků, musel být interval posílání požadavků snížen tak, aby byl co nejkratší. Doba mezi jednotlivými požadavky je tedy nastavena na 15 sekund.

Pro ovládání relé musel být vytvořen speciální postup, aby nekolidovala aktuální hodnota relé v databázi s nastavenou hodnotou ze systému pro sběr a vizualizaci dat. Ovládat relé je možné hned několika způsoby. Prvním způsobem je ovládání relé, které je ve stejné lokální síti jako uživatel, pomocí webového rozhraní dostupného zadáním IP adresy zařízení do prohlížeče. Další možností ovládání relé je pomocí protokolu MQTT. Oba předešlé způsoby ovládání ale neprobíhají přes systém pro sběr a vizualizaci dat. Ovládání relé pomocí systému pro sběr a vizualizace dat je možné ze stránky *dashboard.php* a *devices.php*, kde se po správném vložení zařízení zobrazí blok s ovládacími prvky viz Obrázek 8.



Obrázek 8 Ovládací prvek relé akčního členu

Princip ovládání relé akčních členů spočívá v použití databázové tabulky *rly_control* jako prostředníka. Aktuální stav relé se průběžně ukládá do datových tabulek stejně jako je tomu v případě senzorů BME a PBX. V případě, že uživatel změní stav relé pomocí ovládacích prvků zobrazených na obrázku 8, dojde ke změně hodnoty v tabulce *rly_control*.

Tabulka 7 Hlavička tabulky rly_control

Název sloupce	ID	Rel_SN	Rel_ID	CH1	CH2
Popis	ID záznamu	SN relé	ID relé	Kanál 1	Kanál 2

Tabulka *rly_control* obsahuje hodnoty nutné k identifikaci zařízení a ID relé je pomocí relace propojeno s ID v tabulce *Devices*. Dále se v tabulce nachází číslování neboli ID řádků, sloupec se sériovým číslem relé pro přehlednost při správě tabulky a datové kanály CH1 a CH2.

Datové kanály CH1 a CH2 tabulky *rly_control* mohou nabývat následujících hodnot:

- **0** – příští hodnota relé je nastavena na rozepnuto,
- **1** – příští hodnota relé je nastavena na sepnuto,
- **2** – nastaví relé na vykonání jednoho pulsu HIGH o délce 1 s,
- **3** – nastaví relé na vykonání jednoho pulsu LOW o délce 1 s,
- **255** – není nastavena žádná hodnota pro změnu stavu relé.

Postup zpracování příští hodnoty relé a její odeslání do relé akčního členu RLY je zobrazen na obrázku 7 v bloku „alt RLY“. V případě, že se jedná o zařízení se sériovým číslem začínajícím na RLY, stránka *in.php* odešle požadavek na získání hodnot z tabulky *rly_control,* odkud ze řádku s odpovídajícím ID relé vyčte data ze sloupců CH1 a CH2 pro příští stav relé. Takto získaná data se poté jednoduchou logikou zpracovávají na výsledný textový řetězec, který má podobný formát jako HTTP GET požadavek.

```
if      ($rlyrow['rel1'] == 1)  {$answ_rel1 = "rel1=1&";}
else if ($rlyrow['rel1'] == 0)  {$answ_rel1 = "rel1=0&";}
else if ($rlyrow['rel1'] == 2)  {$answ_rel1 = "rel1=1&del1=0&dur1=1";}
else if ($rlyrow['rel1'] == 3)  {$answ_rel1 = "rel1=0&del1=0&dur1=1";}
else                            {$answ_rel1 = "";}
```

Předchozí úryvek kódu se stará o vytvoření odpovědi pro první kanál relé akčního členu. Výsledná odpověď poté může mít následující tvar:

```
200?rel1=1&del1=0&dur1=1
```

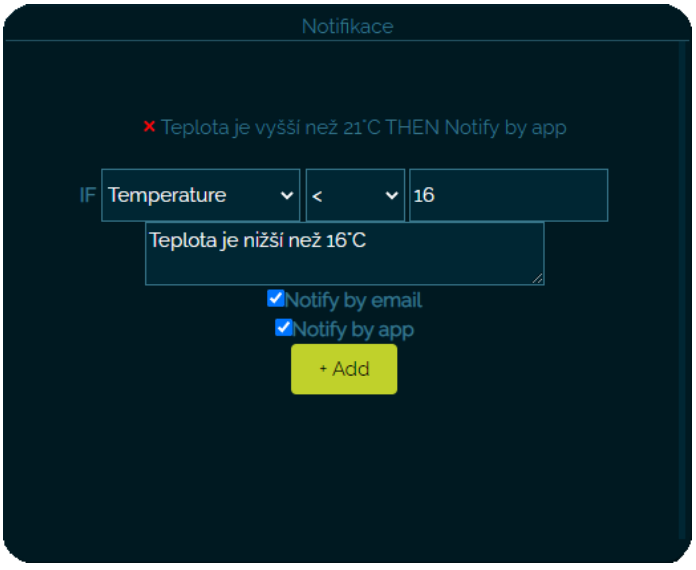
Kde číslice 200 označuje alternativu k stavovému kódu HTTP a metaproměnná *rel1* zde udává, že se jedná o hodnotu pro první kanál relé. Metaproměnná *rel1* obsahuje hodnotu 1, tudíž se bude relé nastavovat na hodnotu 1 na dobu z metaproměnné *dur1 = 1* (podle anglického slova duration – doba trvání v sekundách). Metaproměnná *del1=0* vychází z anglického slova delay, tedy zpoždění, za jak dlouho od přijetí požadavku se impuls provede. Po odeslání odpovědi do zařízení se hodnoty CH1 a CH2 v tabulce *rly_control* přepíší na hodnoty 255, tzn. není nastavena žádná hodnota pro změnu stavu relé.

5.5 Oznámení

Po přijetí každého HTTP POST dotazu a jeho následné validaci, stránka *in.php* mimo jiné také odesílá další HTTP POST požadavek na stránku *eventhandler.php.* Varianta s odesíláním nového požadavku na další stránku byla zvolena kvůli co největšímu odlehčení vstupní stránky *in.php.* Webová stránka *in.php* totiž zpracovává mnoho požadavků na databázi, a po celou dobu jejího vykonávání zůstává otevřené spojení mezi serverem a zařízením. S vyšší komplexitou skriptů na stránce *in.php* se prodlužuje doba jejího zpracování a v některých případech docházelo k vypršení časového limitu, po který zařízení čekala na odpověď od serveru.

Varianta se zpracováním oznámení na externí stránce *eventhandler.php* ušetří mnoho času a spojení je tak možné dříve ukončit a uvolnit prostředky serveru pro další dotazy. Aby bylo zpracování oznámení pro stránku *in.php* ještě efektivnější, odesílá se POST požadavek bez čekání na odpověď od stránky *eventhandler.php*. POST požadavek pro zpracování oznámení, odesílaný funkcí *curl,* obsahuje pouze jednotlivé signály ze zařízení a sériové číslo a jméno zařízení. Postup zpracování oznámení je zobrazen ve zjednodušené formě na obrázku 7.

Pro vytvoření oznámení je možné využít formuláře na stránce zařízení, kde si uživatel může vybrat z několika přednastavených symbolů matematické logiky. Formulář se skládá z rozevíracího seznamu obsahujícího všechny dostupné hodnoty k porovnávání, dále obsahuje rozevírací seznam symbolů matematické logiky (<, <=, >, >=, =, !=), pole pro vepsání srovnávané hodnoty a vlastní zprávu oznámení viz Obrázek 9. Před stisknutím tlačítka *Přidat* (v době pořízení obrázku ještě nebyl kompletně dokončený překlad) si uživatel může vybrat jakým způsobem mu má být oznámení doručeno. Po stisknutí tlačítka *Přidat* se zadané hodnoty uloží do databáze do tabulky *notifications* společně s informacemi o uživateli, který oznámení vytvořil.



Obrázek 9 Formulář pro vytvoření a správu oznámení

Webová stránka *eventhandler.php* nemá při spuštění žádný výstup pro uživatele, po otevření obsahuje pouze prázdnou stránku, protože obsahuje pouze PHP skripty pro zpracování oznámení. Stránka *eventhandler.php* je určena pouze pro přijímání HTTP POST požadavků, podle kterých vyhledává v databázi, jestli dotazované zařízení existuje a jestli je pro něj dostupné oznámení. Jestli je oznámení vytvořené systém zjišťuje pomocí následujícího dotazu na databázovou tabulku *notifications.*

```
$stmt = $link->prepare("SELECT *, notifications.id
        AS n_id
        FROM notifications
        INNER JOIN User
        ON User.id=notifications.userid
        WHERE notifications.dev_id=?"
        );
```

V případě, že je pro ID tohoto zařízení v databázi vytvořen záznam se správnými parametry, potom je záznam odeslán jako odpověď na dotaz z úryvku výše. Pokud ale pro dané ID není v tabulce žádný řádek, není oznámení vytvořené a program končí.

Tabulka 8 Hlavička tabulky notifications

Název sloupce	ID	D_ID	U_ID	sig	operator
Popis	ID záznamu	ID zařízení	ID uživatele	Číslo signálu	Symbol porovnání
Název sloupce	trig	mail	app	custmsg	
Popis	Prahová hodnota	Oznámení emailem	Oznámení aplikací	Vlastní text k oznámení	

Tabulka 8 popisuje názvy jednotlivých sloupců v databázové tabulce *notifications*. Tabulka *notifications* využívá databázových relací na propojení s tabulkou *Devices* a *User*, pomocí kterých systém při jednom dotazu může získat hodnoty i z propojených tabulek.

Pro detekování překročení prahových hodnot musí systém znát aktuální hodnotu signálu a předposlední hodnotu signálu, které porovnává zadaným operátorem. Aktuální hodnotu systém přijímá při POST požadavku a předposlední hodnotu si vyčítá z databáze následujícím příkazem vloženým do PHP funkce mysqli.

```
$sql_predposledni = "   SELECT serverdatetime, sig".$sig_nr."
                        FROM Device_data_".$sn."
                        ORDER BY serverdatetime
                        DESC LIMIT 1, 1";
```

Po získání dvou posledních po sobě jdoucích hodnot signálu systém testuje překročení prahové hodnoty pro daný operátor.

```
if ($rowss['operator']=='>'){
        if (($predp_val<$trig_val)&&($latest_val>$trig_val)){
                notify_sender($send_email,$send_app,$rowss['email'],$sig_nr,$rowss['operator'],
                        $trig_val,$_POST['name'],$sn,$cust_message
                        );
        }
} …
```

Předchozí úryvek kódu se stará o porovnání předposlední a poslední hodnoty signálu daného zařízení pro matematický symbol „>“. Podobný tvar mají i všechny ostatní jmenované podmínky. V případě, že je splněna podmínka, volá se funkce *notify_sender()* se všemi potřebnými parametry. Funkce se větví podle toho, je-li vybráno odeslání oznámení pomocí emailu nebo aplikace Firebase Cloud Messaging.

5.5.1 Oznámení emailem

Pro odesílání oznámení emailem se používá aplikace třetí strany nazývaná PHPMailer. Skript aplikace PHPMailer napsaný v jazyce PHP je jen přiložený jako soubor *PHPMailerAutoload.php* mezi zdrojové soubory systému pro sběr a vizualizaci dat. Pro odeslání emailu stačí pouze připojit jmenovaný soubor jako knihovnu do zdrojového kódu a vytvořit krátký blok kódu s důležitými náležitostmi viz následující úryvek:

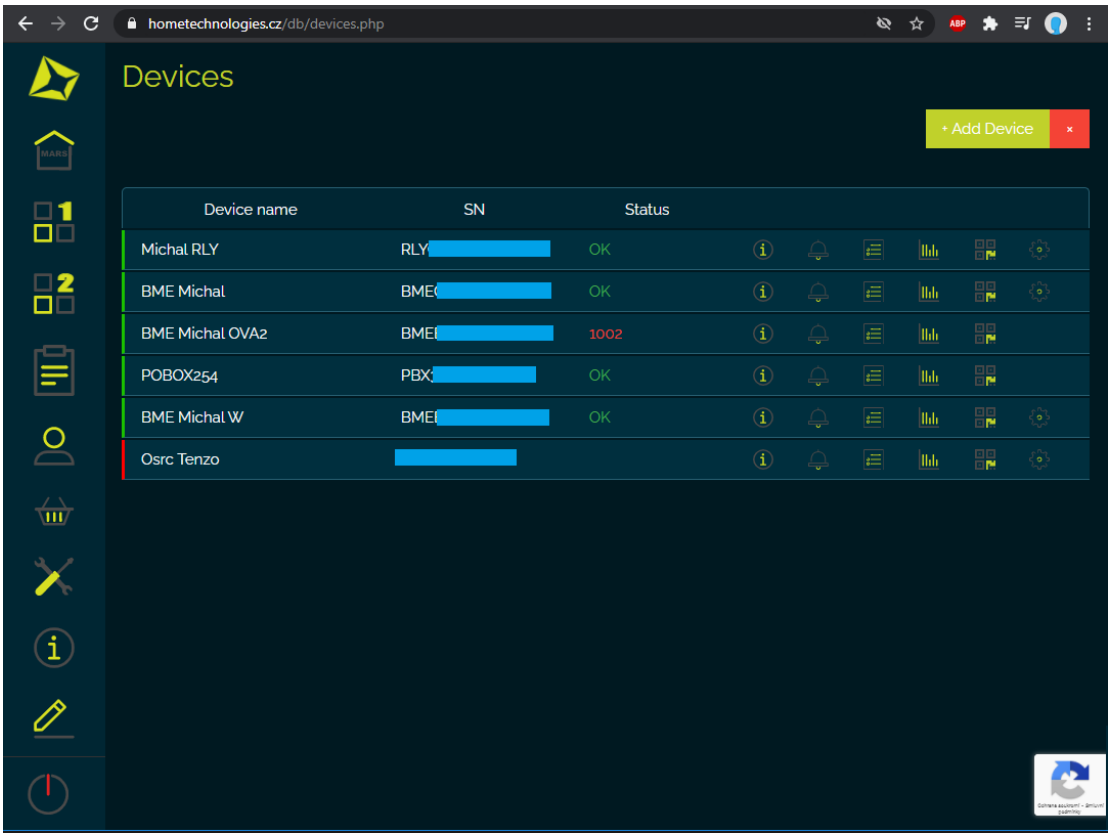
```
require 'phpmailer/PHPMailerAutoload.php';
$mail = new PHPMailer;
$mail->CharSet = "UTF-8";
$mail->isSMTP(true);
$mail->SMTPAuth = true;
$mail->Host = 'smtp-#######.m59.wedos.net';
$mail->Port = 587;
$mail->Username = 'noreply@hometechnologies.cz';
$mail->Password = '############';
$mail->SMTPDebug= 0;
$mail->Debugoutput = 'html';
$mail->setFrom("noreply@hometechnologies.cz", "Home Technologies");
$mail->Subject = 'Home Technologies Notification';
if (!empty($custom_message)){
      $mail->msgHTML($custom_message);
}else{
      $mail->msgHTML("Zpráva emailu");
}
$mail->addAddress($mailaddr);
$mail->SMTPSecure = "tls";
if(!$mail->send()) {
      echo 'Message could not be sent.<br>';
      echo 'Mailer Error: ' . $mail->ErrorInfo;
} else {
      echo 'Message has been sent<br>';
}
```

5.5.2 Firebase Cloud Messaging

Firebase Cloud Messaging je aplikace, která výrazně zjednodušuje zasílání oznámení do mobilních zařízení. Pokud by se každá aplikace například v mobilním telefonu opakovaně doptávala serverů na nová oznámení, způsobovalo by to pro telefon velké zatížení procesoru a spotřebu dat. Firebase Cloud Messaging – dále jen FCM, funguje jako prostředník mezi odesílateli zprávy a klienty, které FCM podporují. Systém pro sběr a vizualizaci dat využívá FCM prostřednictvím skriptů třetích stran na které pouze odesílá HTTP POST požadavky s potřebnými daty.

5.6 Správce zařízení

Jednou z nejdůležitějších součástí systému pro sběr a vizualizaci dat, je stránka *devices.php,* která se stará o správu zařízení a jejich propojení s uživatelskými účty. Po registraci uživatele nejsou k uživatelskému účtu připojena žádná zařízení, jež by bylo možné spravovat. Z toho důvodu byla vytvořena stránka Správce zařízení, na obrázku 10 je vyobrazena ještě v anglickém jazyce. Pomocí tlačítka *Add Device* si uživatel může přidat výrobek do seznamu svých zařízení. Pro přidání musí zadat sériové číslo zařízení, které slouží jako jedinečný identifikátor. Přidávání zařízení do seznamu probíhá přes jednoduchý HTML formulář, který zadané informace odesílá pomocí HTTP POST požadavku zpět na stránku *devices.php*. Stránka se poté znovu načte, nyní s parametry z předešlého POST požadavku, které nejprve ukládá do databáze. Po uložení hodnot z POST požadavku do databáze se stránka znovu dotazuje na databázi na již aktualizovaný seznam zařízení, který vypíše.



Obrázek 10 Stránka správce zařízení (Devices)

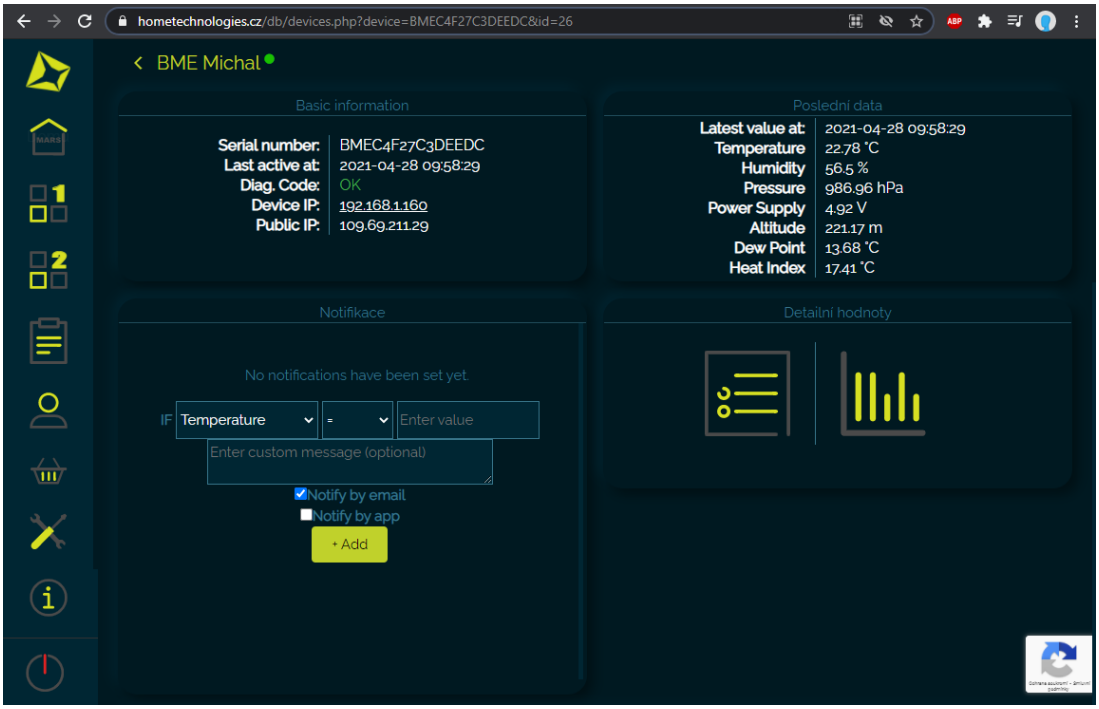
Stránka Správce zařízení (Devices) byla úplně první vyvíjenou webovou stránkou z diplomové práce, jelikož obsahuje výpis všech dat ze senzorů. Během vývoje od roku 2019 již stránka měla mnoho grafických podob a funkcí. První varianty stránky Správce zařízení obsahovaly ve vysouvacích oknech i grafy a tabulky naměřených dat za posledních 24 hodin. To se postupem času začalo projevovat jako velmi nevhodné řešení, protože s přibývajícím počtem zařízení v seznamu se výrazně prodlužovala doba načítání stránky. Současná varianta na hlavní stránce, viz Obrázek 10, obsahuje vysouvací okna pouze pro ovládací prvky, které jsou vyladěny tak, aby jejich dopad na délku nahrávání stránky byl co nejmenší. Pro zobrazení grafů a tabulek se otevírá nové okno, kde se načítají pouze data pro konkrétní senzor za zvolený časový úsek. Jednou z mnoha funkcí Správce zařízení je také rozpoznání, které ze zařízení v seznamu je ve stejné lokální síti jako počítač, na kterém se stránka načetla. Pokud je zařízení ve stejné síti, zobrazí se u něj v seznamu ikonka ozubeného kola, která odkazuje přímo na IP adresu daného zařízení.

Na hlavní stránce Správce zařízení se nachází všechny základní informace o každém přidaném zařízení, jako jsou stav, jméno, sériové číslo a diagnostický chybový kód. Po otevření vysouvacích oken se uživatel dostane k dalším informacím o zařízení, například IP adresy, čas posledního připojení zařízení, zobrazení a nastavení oznámení a podobně.

Kliknutím na libovolné zařízení v seznamu je uživatel přesměrován na stránku daného zařízení, kde se nachází více podrobných informací o zařízení. Stránky jednotlivých zařízení jsou také dostupné ze souboru *devices.php*, ale s přidanými parametry pomocí HTTP GET metaproměnných. Adresa stránky zařízení pak může mít následující tvar:

```
https://hometechnologies.cz/db/devices.php?device=BMEC4F27C3DEEDC&id=26
```

kde je adresa na stránku *devices.php* rozšířena o metaproměnné udávající sériové číslo a ID daného zařízení. Výsledná stránka konkrétního zařízení je zobrazena na obrázku 11.



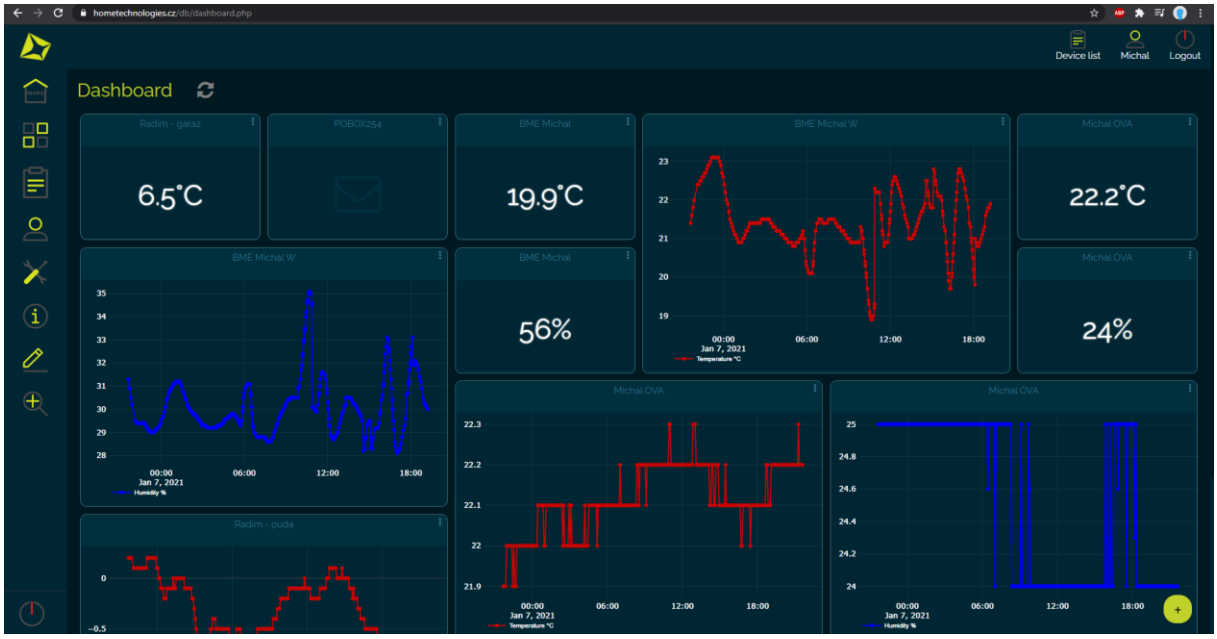
Obrázek 11 Stránka správce zařízení pro jednotlivé zařízení

5.7 Vizualizace dat

Pro zobrazení dat systém využívá především stránku *dashboard.php*, kde si uživatel může postupně přidat různé druhy modulů pro zobrazování hodnot. Stránka *dashboard.php* je pomocí funkce *grid* kaskádových stylů CSS rozdělena na mřížku, která má dynamicky se měnící počet sloupců a řádků v závislosti na velikosti stránky. Moduly pro zobrazování hodnot, viz Obrázek 12, vyplňují jednotlivé buňky mřížky CSS. Na stránku *dashboard.php* je možné vložit moduly několika typů:

1. **Zobrazení hodnoty** – modul velikosti 1x1, který zobrazuje pouze poslední přijatou hodnotu ze zařízení s jednotkami
2. **Graf** velikosti 2x2 s volitelným počtem průběhů stejného typu, například teplot
3. **Graf** velikosti 3x2 se stejnými možnostmi jako graf 2x2
4. **Tabulka** velikosti 2x2
5. **Tabulka** velikosti 3x2

Přidávání modulů je možné pomocí průvodce pro přidání modulu, který se spustí tlačítkem „**+**“ v pravém dolním rohu stránky Dashboard. Průvodce využívá tzv. modální okna, která se zobrazují na stránce dashboard v popředí před obsahem stránky. Průvodce pro přidávání modulů je napsaný v jazyce PHP. Jelikož se jazyk PHP spouští na rozdíl od Javascriptu pouze na straně serveru, není možné pomocí PHP vytvářet plnohodnotné stránky s dynamickým obsahem. Průvodce přidáváním modulů využívá běžné HTML formuláře, které se plní daty získanými z databáze a konfiguračního souboru. Po vyplnění všech polí HTML formuláře a kliknutí na tlačítko „next“ se formulář odešle pomocí HTTP POST požadavku opět na stránku *dashboard.php*, ale s jinými parametry. Stránka po přijetí POST požadavku zobrazí následující okno průvodce přidávání modulů s novými parametry. Pomocí tohoto principu zobrazování obsahu se stránka jeví jako interaktivní.



Obrázek 12 Vizualizace v lednu 2021

5.8 Konfigurační soubor

Pro definici velikostí tabulek, reálných názvů a jednotek jednotlivých signálů systém využívá konfigurační PHP soubor *devconf.php*. Konfigurační soubor *devconf.php* je přiložen do každého souboru, kde je potřeba. Obsahuje několik funkcí a polí, pomocí kterých systém získává konkrétní hodnoty. Příkladem použití konfiguračního souboru je definice velikostí tabulek, které jsou vytvářeny po připojení nového zařízení k systému pro sběr a vizualizaci dat. Následující funkce *DeviceConfigIn()* po zadání sériového čísla vrací počet datových signálů, které dané zařízení používá.

```
function DeviceConfigIn($sn)
{
        if     (like($sn,'BME'))   { $nr_of_sigs = 7; }
        elseif (like($sn,'BMP'))   { $nr_of_sigs = 4; }
        elseif (like($sn,'WTH'))   { $nr_of_sigs = 4; }
        elseif (like($sn,'PBX'))   { $nr_of_sigs = 4; }
        elseif (like($sn,'RLY'))   { $nr_of_sigs = 3; }
        elseif (like($sn,'NCR'))   { $nr_of_sigs = 4; }
        elseif (like($sn,'VSB'))   { $nr_of_sigs = 4; }
        else                       { $nr_of_sigs = 1; }
        return $nr_of_sigs;
}
```

Na podobném principu jako výše zobrazená funkce funguje i další funkce konfiguračního souboru *DeviceErrorNr(),* která při vstupním parametru obsahujícím diagnostický kód zařízení nastaví návratovou hodnotu na textový řetězec obsahující popis daného diagnostického kódu.

Nejdůležitější funkcí konfiguračního souboru *devconf.php* je funkce *DeviceConfig()* obsahující pole s názvy signálů jednotlivých zařízení, jejich jednotkami a barevným označením.

```
function DeviceConfig($sn)
{
       if (like($sn,'BME'))
       {
               $graphset = array();
               $graphset[0] = array(
                       "name" => 'Temperature',
                       "unit" => '°C',
                       "color" => 'rgb(200, 5, 5)'
               );

               $graphset[1] = array(
                       "name" => 'Humidity',
                       "unit" => '%',
                       "color" => 'rgb(0, 0, 255)'
               );
…
}
```

Při vykreslování všech signálů z databáze tak systému postačí znát pouze sériové číslo zařízení a číslo signálu a pomocí funkce *DeviceConfig()* získá veškeré údaje o hodnotě. Například volání funkce:

```
DeviceConfig(<sn>)[<číslo_signálu>]["name"]
```

vrátí název hodnoty pro daný signál. Konfigurační soubor *devconf.php* využívají téměř všechny stránky webu.

5.9 Další součásti systému

Mezi další, méně podstatné součásti systému pro sběr a vizualizaci dat, je možné zmínit například REST API dostupné na adrese api.hometechnologies.cz, pomocí kterého je možné po zadání přístupového tokenu získávat některá data z databáze ve formátu JSON.

Další součástí systému je také administrátorské rozhraní, přístupné pouze administrátorům webu. V administrátorském rozhraní jsou umístěné interní dokumenty, data z Google analytics, testovací HTML formuláře, informace o stavu databáze, KPI a servisní stránka pro zjištění všech dat o vybraném zařízení. V neposlední řadě je možné zmínit také stránku *user.php* dostupnou pouze přihlášeným uživatelům. Na stránce *user.php* jsou vypsány informace o přihlášeném uživateli uložené v databázové tabulce *User*.

6 Testování a verifikace funkcí a parametrů systému

Tato kapitola se zabývá ověřením správnosti funkce vytvořené webové aplikace a ukládání dat do SQL databáze. Další část kapitoly pojednává o rychlosti zpracování některých skriptů webu a srovnání odhadů velikosti databáze s reálnou velikostí po určité době používání.

6.1 Srovnání odhadu velikosti databáze se skutečností

Kapitola 4.3 se zabývala odhadem velikosti databáze. V této kapitole se prokáže, nakolik byl odhad velikosti databáze přesný. Databáze v současné době (duben 2021) obsahuje 206 zařízení a některá z nich komunikují s databází více než rok.

Tabulka 9 Shrnutí stavu databáze na konci dubna 2021

Počet aktivních zařízení	206
Počet online zařízení	35
Počet zařízení připojených naposledy před méně než měsícem	27
Počet neaktivních zařízení déle než měsíc	144
Počet uživatelů	15
Velikost databáze	269 MB

Databáze systému pro sběr a vizualizaci dat je spuštěná již více než rok, a některá testovací zařízení jsou nastavena tak, aby co nejrychleji plnila databázi. Automatické odmazávání starých dat ještě není spuštěno, a tak databáze obsahuje velké množství řádků.

Databáze MariaDB spravovaná přes prostředí phpMyAdmin obsahuje kromě datové části i tzv. *information_schema,* ve kterém se nachází seznamy tabulek i s jejich velikostmi a vlastnosti databáze. Pro srovnání odhadu a reality bylo vybráno zařízení BME se sériovým číslem BMEC4F27C3DEEDC, které je připojené k databázi od 22. března 2020. Zmíněné BME má v datové tabulce již 118315 řádků záznamů s velikostí 7,5 MB. Součást databáze nazývaná *information_schema* obsahuje podrobnější informace o BMEC4F27C3DEEDC, viz Tabulka 10.

Tabulka 10 Informace o tabulce zařízení BME

	Jméno zařízení	**Počet řádků**	**Velikost v DB**	**Průměrná velikost řádku**
Reálná tabulka	BMEC4F27C3DEEDC	118315	7 880 704 B	66 B
Odhad (za rok)	-	105120	4 204 800 B	40 B

Z dat získaných z *information_schema* databáze systému pro sběr a vizualizaci dat zobrazených v tabulce 10 vyplývá, že průměrná velikost řádku tabulky zařízení je 66 B. Při odhadu velikosti databáze v kapitole 4.3 byla vypočtena předpokládaná velikost řádku 40 B, viz Tabulka 4. Reálná tabulka je výrazně větší, než se předpokládalo.

6.2 Měření délky vykonávání skriptů

Jako nejlepší způsob, jak ověřit funkčnost a rychlost zpracování dat, se jeví testování stránky *in.php*. Stránka *in.php* se stará o přijímání veškerých dat do systému a spouštění všech skriptů pro zpracování dat. Funkčnost této stránky lze považovat za ověřenou, jelikož bez podstatných změn za dobu svého fungování vyplnila přibližně 4 900 000 řádků tabulek v databázi.

Pro testování rychlosti stránky *in.php* byla navrhnuta metoda měření přesného času při přijetí HTTP POST požadavku, a také na konci skriptu stránky *in.php.* Po spuštění stránky POST požadavkem se na úplném začátku skriptu zapíše do proměnné *$starttime* čas v mikrosekundách.

```
$starttime = microtime(true);
```

Na konci skriptu se stejným způsobem zaznamená čas do proměnné *$endtime.*

```
$endtime = microtime(true);
```

Jelikož stránka *in.php* nemá žádný grafický výstup a pro správnou funkci sbírání dat ze senzorů musí odpovídat pouze stavovými kódy HTTP, je výsledný čas zapsaný do textového souboru. Pro přehlednost se do textového souboru pomocí následujícího příkazu zapíše i čas měření a sériové číslo zařízení.

```
file_put_contents("in_debug.txt",
            date("d.m.Y H:i:s")." - SN: ".$sn."
             - Čas: ".(($endtime - $starttime)*1000)."\n",FILE_APPEND
            );
```

Čas v mikrosekundách se v úryvku kódu přepočítává na milisekundy. Výsledek testování v textovém souboru *in_debug.txt* vypadá následovně:

```
29.04.2021 08:52:43 - SN: RLYG##########6 - Čas: 989.53914642334
29.04.2021 08:52:43 - SN: BME9##########6 - Čas: 976.60994529724
29.04.2021 08:52:43 - SN: - Čas: 1.5490055084229
29.04.2021 08:52:44 - SN: RLYE##########F - Čas: 516.64996147156
29.04.2021 08:52:44 - SN: BMEB##########6 - Čas: 544.28505897522
29.04.2021 08:52:51 - SN: RLYC##########F - Čas: 458.97603034973
29.04.2021 08:52:52 - SN: BMEG##########C - Čas: 454.86783981323
29.04.2021 08:52:53 - SN: BMEC##########F - Čas: 492.12980270386
29.04.2021 08:52:53 - SN: RLY9##########5 - Čas: 459.17987823486
29.04.2021 08:52:55 - SN: BMEB##########A - Čas: 442.78812408447
29.04.2021 08:52:57 - SN: RLY9##########6 - Čas: 753.25703620911
29.04.2021 08:52:58 - SN: RLYG##########6 - Čas: 781.92400932312
```

Z důvodu zabezpečení uživatelských zařízení, jsou sériová čísla v úryvku kódu skryta. Čas provádění jednotlivých požadavků se pohybuje v rozmezí cca 400 až 1000 ms, což je pro zpracovávání skriptů velmi dlouhá doba. V případě neplatného požadavku, bez zadaného sériového čísla je, čas zpracování skriptu stránky pouze 1,5 ms. Velmi dlouhá doba zpracování skriptů na stránce *in.php* může být způsobena velkým zatížením databáze. Databáze obsahuje i jiné tabulky, které nejsou součástí diplomové práce a s velkou pravděpodobností mohou být příčinou pomalé odezvy databáze. Příkladem tabulky zatěžující databázi může být tabulka *translation*, která obsahuje překlady webu, a pro každý textový řetězec na webu je pak dotazována databáze. Například při testování stránky *db/index.php* o velikosti pouze 5 kB se projevuje doba čekání na odpověď serveru (TTFB) až 160 ms, přičemž tato stránka obsahuje jen prostý text a obrázky. Zpracování překladů webu pomocí tabulky *translation* není součástí práce, ani dílem autora práce.

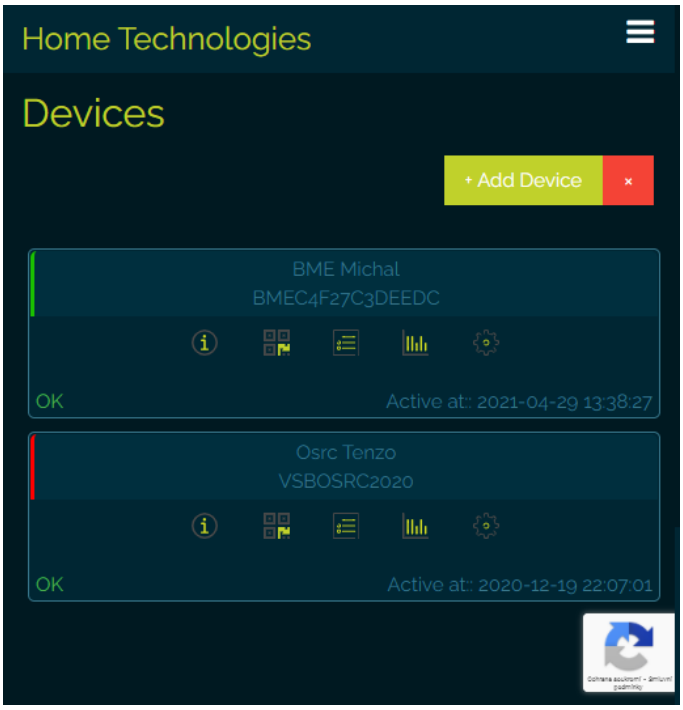
6.3 Ověření funkčnosti uživatelského rozhraní

Pro testování systému pro sběr a vizualizaci dat byl vytvořen speciální účet, který může použít kdokoliv. Údaje pro přihlášení k tomuto účtu jsou:

Email: cep0030@vsb.cz
Heslo: vsbtuo

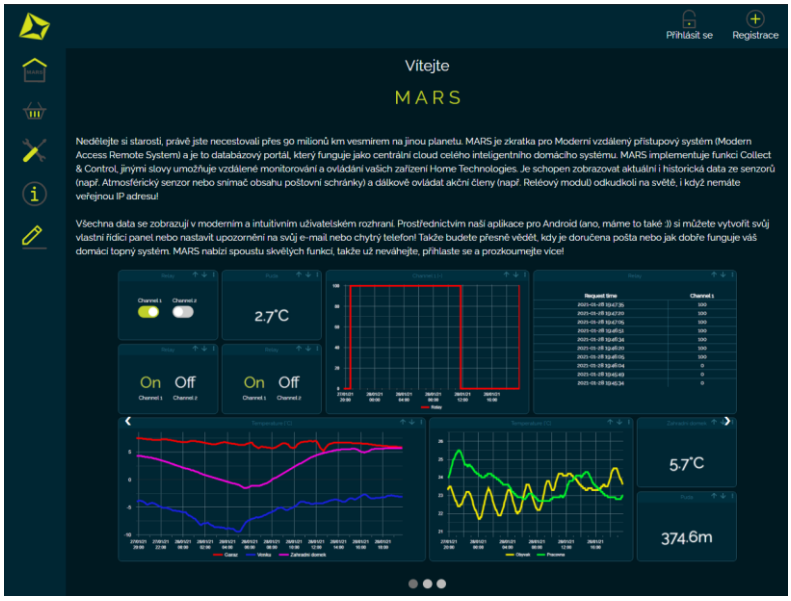
Každá ze stránek systému byla testována na rychlost odezvy pomocí sady nástrojů Chrome DevTools, které jsou vestavěny do prohlížečů Chrome. Nástroje Chrome DevTools umožňují zobrazení průběhů načítání webových stránek, jejich velikosti a případné chyby.

Dále byly všechny webové stránky testovány na různých webových prohlížečích, a to Google Chrome, Mozilla Firefox, Microsoft Edge a Microsoft Internet Explorer. Bohužel v případě Internet Exploreru se na stránkách projevují různé chyby. Vzhledem k tomu, že výrobce prohlížeče společnost Microsoft oznámila ukončení podpory prohlížeče Internet Explorer ke 30. listopadu 2020, nebude tento prohlížeč podporovat ani web společnosti Home Technologies. Jelikož jsou všechny stránky responzivní tzn. optimalizované i pro menší zařízení, musely být testovány také na mobilních zařízeních a různých prohlížečích. O změnu velikosti obsahu webu se starají zejména kaskádové styly CSS.



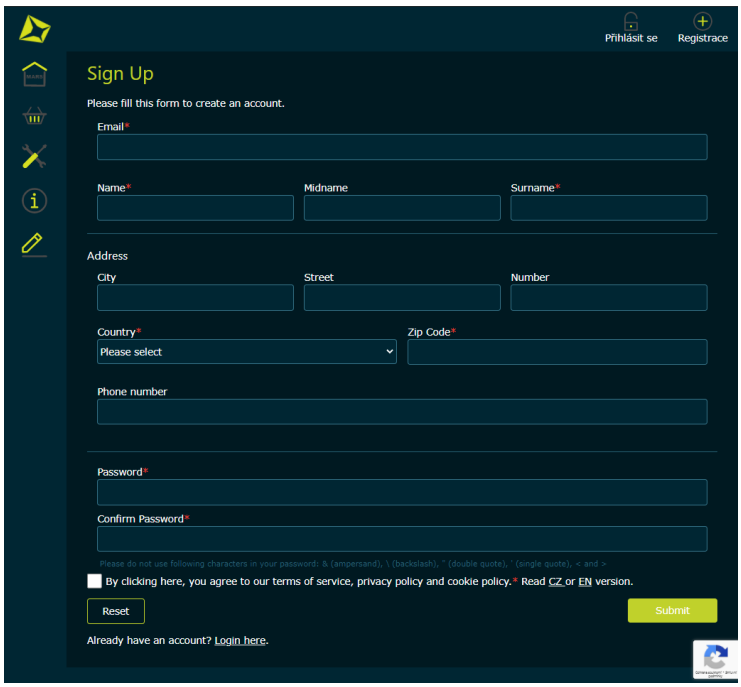
Obrázek 13 Příklad zobrazení stránky správce zařízení na malých zařízeních

Po spuštění webu zadáním adresy *www.hometechnologies.cz/db* se uživatel dostane na úvodní stránku systému pro sběr a vizualizaci dat, viz Obrázek 14. Stránky jsou navrhnuty v tmavě modré barvě se žlutými prvky. Na levé straně všech stránek se nachází menu, které slouží jako rozcestník na hlavní stránku společnosti, úvodní stránku systému pro vizualizaci dat, e-shop, stránku s nástroji a manuály ke stažení, stránku s často kladenými dotazy a kontaktní formulář pro získání technické podpory. Po přihlášení se menu rozšíří o další tlačítka umožňující přístup k datům, správě senzorů a stránce uživatele.



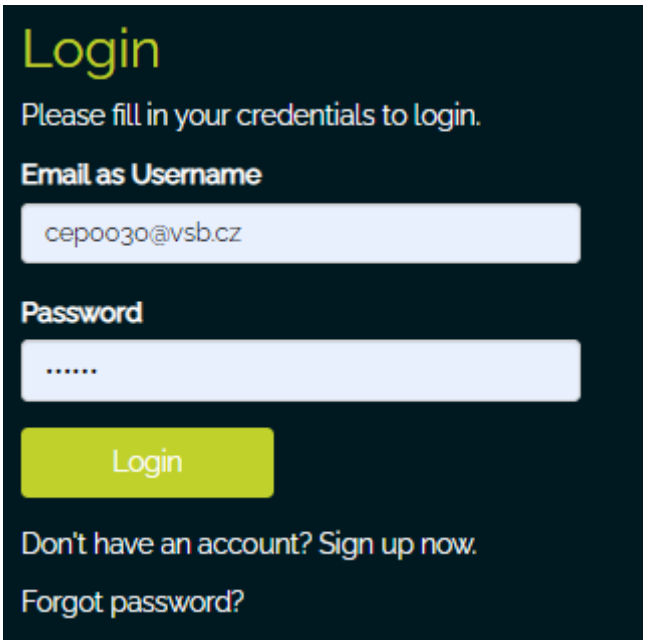
Obrázek 14 Úvodní stránka webu Home Technologies

Aby se uživatel mohl přihlásit, musí se nejprve registrovat kliknutím na tlačítko *Registrace* v pravém horním rohu webu, viz Obrázek 15.



Obrázek 15 Registrace nového uživatele

Na stránce *Registrace* (Sign Up) se nachází formulář k zadání informací o uživateli. Povinná pole jsou zvýrazněna červenými hvězdičkami. V případě, že některá z povinných polí nejsou vyplněna správně, je o tom uživatel informován a může hodnoty upravit. Po úspěšné registraci se uživatel může přihlásit, viz Obrázek 16.



Obrázek 16 Přihlašovací stránka

Stránky s poli k vyplnění jsou chráněny pomocí služby Google reCAPTCHA v3 tzv. boty – automaty, které se snaží web napadnout a zneužít. Po přihlášení je uživatel přesměrován na stránku Dashboard, viz Obrázek 12, kde se nachází grafy, tabulky a další moduly pro zobrazení hodnot nebo ovládání akčních členů. Zobrazovací moduly je možné přidávat pomocí průvodce, viz Obrázek 17.

Obrázek 17 Průvodce pro přidání modulů na stránku Dashboard

7 Zhodnocení dosažených výsledků

Vytvořený systém pro sběr a vizualizaci dat funguje podle základních specifikací společnosti Home Technologies. Mezi základní specifika patřilo vytvoření systému, který bude schopen přijímat a zpracovávat data přijatá ze zařízení společnosti. Tato část zadání je splněna dle požadavků, jelikož systém od svého neveřejného spuštění v dubnu 2020 pracuje bez větších obtíží. Přibližně od ledna 2021 je systém pro sběr a vizualizaci dat veřejně dostupný nejen pro zákazníky společnosti Home Technologies.

Při práci na systému pro sběr a vizualizaci dat bylo nutné nastudovat velké množství dokumentů a návodů. V první řadě bylo nutné vybrat programovací jazyk, ve kterém bude celý systém napsán. Z mnoha navrhnutých programovacích jazyků jako jsou například React, ASP.NET a PHP byl vybrán skriptovací jazyk PHP, a to právě kvůli jeho velkému rozšíření, podpoře mnoha nadstaveb a v neposlední řadě kvůli znalostem jazyka z předešlých projektů. Dalším bodem pro výběr skriptovacího jazyka PHP byl i fakt, že nejlevnější webhostingy podporují nejčastěji jen základní programovací jazyky jako je PHP.

Výběr databáze pro uchovávání dat probíhal obdobně jako výběr programovacího jazyka. Nejprve byly shrnuty všechny výhody a nevýhody navrhnutých databází, ale nakonec možnost výběru databáze odpadla, jelikož k webhostingu od společnosti Wedos je v ceně i databáze MariaDB.

Po zřízení domény Hometechnologies.cz a propojení s webhostingem a databází, započal vývoj samotného webu. Mezi první kroky při vývoji webových stránek patřilo zkoušení různých druhů přenosu dat mezi zařízením a serverem. Jako nejvýhodnější varianta komunikace se projevil přenos dat pomocí HTTP POST požadavků a jejich následnému uložení do databáze MariaDB. Spolu se zvyšujícím se počtem připojených zařízení proběhlo ještě mnoho úprav struktury databáze až do dnešní podoby. V současnosti jsou v databázi tabulky pro uchování seznamu všech zařízení spolu s informacemi o nich, tabulka se seznamem uživatelů a datové tabulky pro každé zařízení. Po prvním připojení zařízení k systému se automaticky vytvoří všechny potřebné tabulky a záznamy do databáze. Není proto nutné do chodu systému nijak výrazně zasahovat.

Jelikož je systém pro sběr a vizualizaci dat popsaný v této práci vytvořený pro firmu Home Technologies, jeho vývoj odevzdáním práce nekončí, ale bude nadále pokračovat. V současné době se pracuje na mnoha dalších rozšíření systému, práci s daty a komunikaci mezi zařízeními. Za zmínku stojí také to, že celý web systému pro sběr dat je navržen tak, aby odpovídal posledním trendům v oblasti web designu.

Závěr

Cílem diplomové práce bylo provést popis problematiky vývoje systému pro sběr a vizualizaci senzorických dat pro internet věcí a následně systém realizovat.

První část práce byla zaměřena na rozbor problematiky sběru a zpracování dat ze senzorů pro internet věcí. V kapitole byl představen trend současnosti internet věcí a popsány různé přenosové technologie, které využívá.

V druhé části diplomové práce byla provedena rešerše dostupných systémů pro ukládání dat ze senzorů pro internet věcí. Byly zde popsány hlavní rozdíly mezi SQL a NoSQL databázemi a další dostupné systémy pro ukládání dat.

Následující část práce se zabývala návrhem systému sběru a vizualizace dat, kde byl vylíčen postup výběru webhostingu a databáze. Při výběru webhostingu byly zohledněny požadavky společnosti Home Technologies s.r.o., a proto byl zvolen webhosting od poskytovatele Wedos, avšak bez možnosti výběru databáze. Byla použita databáze MariaDB, protože je jedinou nabízenou databází zdarma k webhostingu. Dále zde byl proveden návrh struktury databáze a výběr názvů a datových typů jednotlivých sloupců tabulek.

V další kapitole byl systém sběru a vizualizace dat realizován. Součástí kapitoly byla charakteristika jednotlivých prvků webového rozhraní, například proces vytvoření webu následovaný popisem zpracování přijatých dat ze zařízení. Byly zde popsány důležité součásti webu pro správu připojených zařízení, zpracování oznámení a vizualizaci shromážděných dat pomocí tabulek a grafů.

Předposlední kapitola diplomové práce je věnována testování a verifikaci funkcí a parametrů systému. Obsahem kapitoly bylo srovnání odhadu velikosti databáze se skutečností a měření délky vykonávání skriptu při přijetí dat ze zařízení.

V závěru práce byly zhodnoceny dosažené výsledky. Po přibližně ročním používání je systém stále funkční a splňuje podmínky zadání společnosti Home Technologies s.r.o. V době odevzdání diplomové práce byl systém zpřístupněn veřejnosti. V té době bylo v systému registrováno 15 uživatelských účtů a celkem 206 zařízení, včetně testovacích výrobků.

Systém byl vytvořen pro firmu Home Technologies, a tak bude jeho vývoj pokračovat i v budoucnu. Je v plánu systém doplnit o průběžné odmazávání starých dat z databáze pro uvolnění místa. Pro uvolnění místa v databázi se bude pravidelně spouštět skript, pomocí softwarového démona CRON, který slouží jako plánovač úloh. Skript by měl redukovat data starší než jeden měsíc tak, aby v databázi zůstal pouze jeden záznam za hodinu.

Mezi další plány do budoucna patří optimalizace rychlosti databáze a skriptů a zvyšování zabezpečení webu proti případnému napadení a zneužití dat.

Literatura

[1] POKORNÝ, Jaroslav a Michal VALENTA. *Databázové systémy.* 2. přepracované vydání. Praha: Česká technika - nakladatelství ČVUT, 2020. Vysokoškolská učebnice. ISBN 978-80-01-06696-6.

[2] LAURENČÍK, Marek. *SQL: podrobný průvodce uživatele.* Praha: Grada Publishing, 2018. Průvodce. ISBN 978-80-271-0774-2.

[3] SKLAR, David. *PHP 7: praktický průvodce nejrozšířenějším skriptovacím jazykem pro web.* Přeložil Jan POKORNÝ. Brno: Zoner Press, 2018. Encyklopedie Zoner Press. ISBN 978-80-7413-363-3.

[4] HILLS, Ted. *NoSQL and SQL Data Modeling: Bringing Together Data, Semantics, and Software.* Basking Ridge: Technics Publications, 2016. 258 s. ISBN 1634621093.

[5] Wi-Fi. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2021 [cit. 2021-1-2]. Dostupné z: https://cs.wikipedia.org/wiki/Wi-Fi

[6] Konektivita do sítě LoRaWan s DNS Marketplace. *DNS Marketplace* [online]. DNS Marketplace, 2020 [cit. 2021-1-2]. Dostupné z: http://marketplace.dns.cz/blog/konektivita-do-site-lorawan/

[7] LOM, Ing. Michal a prof. Ing. Ondřej PŘIBYL, Ph.D. Sítě pro internet věcí v České republice. *TZB-Info*. 2017.

[8] *Sigfox* [online]. Praha, 2020 [cit. 2021-1-3]. Dostupné z: https://sigfox.cz/cs

[9] Co je NB – IoT. *Vodafone* [online]. 2020 [cit. 2021-1-5]. Dostupné z: https://www.vodafone.cz/firmy-a-korporace/internet-veci/nb-iot1/

[10] Wi-Fi 6. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2021 [cit. 2021-2-22]. Dostupné z: https://cs.wikipedia.org/wiki/Wi-Fi_6

[11] MAC adresa. *Wikipedia* [online]. San Francisco, 2020 [cit. 2021-4-3]. Dostupné z: https://cs.wikipedia.org/wiki/MAC_adresa

[12] KLENSIN, J. *RFC 3696: Application Techniques for Checking and Transformation of Names* [online]. The Internet Society, February 2004, (3696), 6 [cit. 2021-4-13]. Dostupné z: https://tools.ietf.org/html/rfc3696

[13] Data Type Storage Requirements. *MariaDB* [online]. 2013 [cit. 2021-4-13]. Dostupné z: https://mariadb.com/kb/en/data-type-storage-requirements/

[14] *Recommendation ITU-T E.164: The international public telecommunication numbering plan.* International Telecommunication Union. Geneva, 2010, (164), 5. Dostupné také z: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.164-201011-I!!PDF-E&type=items

[15] *DB-Engines* [online]. Wien, 2020 [cit. 2021-4-13]. Dostupné z: https://db-engines.com/en

[16] WELLING, Luke a Laura THOMSON. *Mistrovství PHP a MySQL*. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.

[17] *Firebase* [online]. Mountain View, Kalifornie, USA: Google, 2021 [cit. 2021-4-30]. Dostupné z: https://firebase.google.com/

[18] *ThingSpeak* [online]. Natick, Massachusetts , USA, 2021 [cit. 2021-4-30]. Dostupné z: https://thingspeak.com/pages/learn_more

[19] *Microsoft Azure* [online]. Redmond, Washington, USA: Microsoft, 2021 [cit. 2021-4-30]. Dostupné z: https://azure.microsoft.com/cs-cz/

[20] HOLUBOVÁ, Irena, Jiří KOSEK, Karel MINAŘÍK a David NOVÁK. Big Data a NoSQL databáze. Praha: Grada, 2015. Profesionál. ISBN 978-80-247-5466-6.

[21] KADLEC, Roman. Ako zdokonaľovať mobilné a webové aplikácie? Vyskúšajte službu Firebase. *Touchit*. 2016, (9).

[22] *Home Technologies* [online]. Ostrava, 2021 [cit. 2021-4-29]. Dostupné z: http://www.hometechnologies.cz/cz/index.php

Seznam příloh